Large Eddy Simulation of Vertical Axis Wind Turbine for Low Reynolds Number Applications

Matin Komeili

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Mechanical Engineering) at

Concordia University

Montréal, Québec, Canada

September 2016

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Mr. Matin Komeili**

Entitled: **Large Eddy Simulation of Vertical Axis Wind Turbine for Low Reynolds Number Applications**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Mechanical Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Deborah Dysart-Gale

_____ External Examiner
Dr. Éric Laurendeau

_____ External to Program
Dr. Pragasen Pillay

_____ Examiner
Dr. Ali Dolatabadi

_____ Examiner
Dr. Hoi Dick Ng

_____ Thesis Supervisor
Dr. Marius Paraschivoiu

Approved by _____
Martin D. Pugh, Chair
Department of Mechanical and Industrial Engineering

_____ 2016

_____
Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Large Eddy Simulation of Vertical Axis Wind Turbine for Low Reynolds Number Applications

Matin Komeili, Ph.D.

Concordia University, 2016

Research of low to intermediate Reynolds numbers flows $(1000 < Re < 10^6)$ has become essential in the last two decades due to increasing interest in small Unmanned Aerial Vehicles (UAVs), small wind turbines, and exploration of planets such as Mars. Simulation of many of these applications need an accurate prediction of aerodynamic forces within five percent accuracy. Therefore, developing high accurate Computational Fluid Dynamics (CFD) tools is essential to design and optimize these systems.

Simulation of a Vertical Axis Wind Turbine (VAWT) on Mars is considered as the motivating application in this project. It is motivated by current plans of sending humans to Mars in the next two decades, and the need to investigate sustainable way to generate power on the planet.

VAWTs have a simple geometry but the flow structure around the blade is known to be one of the most complex flow in aerodynamics. Separation, vortex shedding and dynamic stall frequently occurs on the turbines' blade. Therefore a tool that accurately simulates the flow around wind turbines, accurately, is needed.

Large eddy simulation has proven to be a reliable turbulence model with the capability of simulation flows with regions of separation and transition to turbulence. Growth of the computational capability along with development of more accurate numerical methods and new advanced LES models has permitted the use of wall-resolved LES.

In the current dissertation, Wall-Adapting Local Eddy-Viscosity (WALE) simulation is used to simulate flow around a vertical axis wind turbine. Using a second-order accurate discretization technique, both in space and time, with a low-dissipative method, enforced by an adjustable upwinding

factor, achieves the required accuracy in Large Eddy Simulation. The proposed approach enables us to accurately predict the shear stress and pressure distribution on the blade. Therefore, dynamic stall location is spotted precisely.

The potential to increase the performance of small VAWTs by using Morphing blades are very promising. An in-house code has been extended to simulation flow around dynamically morphing blades. Therefore, Arbitrary Lagrangian Eulerian (ALE) is used to preserve the second order accuracy of the numerical scheme under a dynamic mesh, and a combination of spring and diffusion methods is used to adjust the mesh dynamically around deforming blades. A morphing blade scenario is presented to show the new capability developed.

Acknowledgements

I would like to thank my thesis supervisor, Professor Marius Paraschivoiu, for his immeasurable patience, support, guidance, and insight. It was my honor working with him during these years, , and I am always grateful to him. I must also thank Professor Paraschivoiu for his prompt review and valuable feedback for this document.

I also would like to thank Compute Canada and Calcul Quebec for providing the computer resources to run our simulations, including Colosse, Guillimin, Mammouth parallle II, and Briare. Specially, I want to thank Bart Oldeman from Guillimin and HuiZhong Lu from Mammouth parallle II.

I also want to thank Bombardier Aerospace and MITACS for having a 9-months internship at Advanced Aerodynamics Department, where I had a wonderful and valuable experience. I appreciate the kindness and help of all people in Advance Aerodynamics Department. I am specially grateful to Dr. Patrice Castonguay, my supervisor, and Dr. Hong Yang, CFD section Chief, for their help, support, and guidance.

My special thanks goes to my friend Dr. Kaveh Mohamed for his valuable and extensive consult during the developing process of our in house code. Definitely without his guidance I would not be able to accomplish my thesis. I am also sincerely grateful to the CFD team at Concordia University during these years. Discussing with them always were helpful to find new solution for my problems, and specially thanks to Gabriel Naccache for providing valuable feedbacks on my thesis draft.

I also thank some of my friends, Roham Mactabi, Bahareh Amirjabbari, Sivash Taheri, Alireza Pazooki and Sahar Alinia for their friendship and support.

Finally, I would like to thank my loving parents, Farideh and Mahmoud Komeili, my sibling Milad, Moin, and Monica Komeili, and my sister-in-law, Hediyeh Mohseni-niya for their unconditional love and support at my most challenging times. I am forever grateful to them. I would like to dedicate my thesis to my family.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

ALE Arbitrary Lagrangian Eulerian

AoA Angle of Attack

AU Astronomical Unit

$BiCGSTAB$ Bi-Conjugate Gradient Stabilized

CFD Computational Fluid Dynamics

CFL Courant-Friedrich-Lewy

$DMST$ Double-multiple Streamtube

DNS Direct Numerical Simulation

$GMRES$ Generalized Minimal Residual

$HAWT$ Horizontal Axis Wind Turbine

JPT Jet Propulsion Laboratory

KH Kelvin-Helmholtz

LES Large Eddy Simulation

MFC Macro-Fiber Composites

$MUSCL$ Monotonic Upstream-Centered Scheme for Conservation Laws

$N-S$ Navier-Stokes

$NACA$ National Advisory Committee for Aeronautics

$NASA$ National Aeronautics and Space Administration

PDE Partial Differential Equations

$PETSc$ Portable Extensible Toolkit for Scientific Computation

$RANS$ Reynolds-Averaged Navier-Stokes

RTG Radioisotope Thermoelectric Generator

SFS Sub-Filter Scale

SGS Sub-Grid Scale

SST Shear Stress Transport

TSR Tip Speed Ratio

TVD Total Variation Diminishing

$URANS$ Unsteady Reynolds-Averaged Navier-Stokes

$VAWT$ Vertical Axis Wind Turbine

$WALE$ Wall Adapting Local Eddy Viscosity

Latin

A_s Sweep Area by a Wind Turbine $[m^2]$

c Speed of Sound $[m/s]$

$C(I)$ Finite Volume Cell

C_D Drag Coefficient

C_d Drag Coefficient

C_L Lift Coefficient

C_l Lift Coefficient

C_p Pressure Coefficient; or Power Coefficient

c_p Specific Energy in Constant Pressure $[(kN.m)/(kg.K)]$

c_v Specific Energy in Constant Volume $[(kN.m)/(kg.K)]$

C_{ex} Expansion-contraction parameter

C_{SM} Modified Smagorinsky Constant

C_S Smagorinsky Constant

C_w WALE Constant

d_{ij} Distance between point i and j

E Total Energy $[J]$; or Young's Modulus $[Pa]$

$E(I)$ Finite Element Cell

e_I Internal Energy $[J]$

h Enthalpy $[J]$

k Thermal Conductivity $[W/(m.K)]$

k_{ij} Stiffness of a spring

M_I Finite Volume Weight Function

N_I Finite Element Weight Function

N_s Normal vector to a surface

P Turbine Power $[W]$

p Pressure $[Pa]$

Pr	Prandtl Number
Pr_t	Turbulent Prandtl Number
Q_i	Heat Flux $[W/m^2]$
q_i	Heat Flux $[W/m^2]$
Q_i^{SGS}	SGS heat flux $[W/m^2]$
R	Specific Gas Constant $[(kN.m)/(kg.K)]$
Re	Reynolds Number
S_{ij}	Shear Rate Tensor $[m/s]$
S	Surface Area
St	Strouhal Number
T	Temperature $[K]$
T	Torque $[N.m]$
t	Time $[s]$
T_c	Sutherland Temperature $[K]$
T_s	Reference Temperature $[K]$
U	Velocity $[m/s]$
u_i	Velocity component $[m/s]$
u_τ	Friction Velocity $[m/s]$
V	Volume of Finite Element Cell
vol	Volume of Finite Volume Cell
x_i	Coordinate Axis

y^+ Dimensionless Wall Distance

Greek

α Angle of Attack; or Upwind Factor

α_{ss} Static Stall Angle of Attack

Δ Filter Size

δ_{ij} Kronecker Delta

ϵ Dissipation rate

η Kolmogorov Length Scale

γ Ratio of Specific Heat; or Diffusion coefficient

κ Power in the edge-stiffness coorelation

λ Tip Speed Ratio

$\mathbf{\Lambda}_{ij}$ Diagonal part of Jacobian Matrix

μ Dynamic Viscosity $[Pa.s]$

μ_s Reference Dynamic Viscosity

μ_{SGS} SGS dynamic Viscosity $[Pa.s]$

ν Kinematic Viscosity $[m^2/s]$

Ω Computational Domain

ω Angular Velocity $[rad/s]$

Ω_{ij} Vorticity $[m/s]$

ϕ Test Function

ρ Density $[kg/m^3]$

σ_{ij} Viscous Stress Tensor $[Pa]$

τ Time Step Ratio

τ_w Wall Shear Stress $[m/s]$

τ_{ij} SGS Shear Stress Tensor $[m/s]$

θ Azimuthal Angle of VAWT; or Wiggle Detector Threshold

Mathematical

$\mathbf{S}$ Vector of Source Fluxes

$\mathcal{D}_j$ SGS Viscous Diffusion

$\mathcal{J}_j$ SGS Turbulence Diffusion

∇ Nabla Operator

∂ Partial Derivative

$\partial C(I)$ Boundary of Cell

$\boldsymbol{A}_{ij}$ Convective Jacobian Matrix

$\boldsymbol{P}_{ij}^{-1}$ Right Primitive Variable Jacobian Matrix

$\boldsymbol{P}_{ij}$ Left Primitive Variable Jacobian Matrix

$\boldsymbol{W}$ Vector of the Conservative Variables

$\vec{n}$ Normal to surface

${}^{c}\boldsymbol{F}$ Vector of the Convective fluxes

${}^{v}\boldsymbol{F}$ Vector of the Viscous fluxes

div Divergence

exp Exponential Function

$G(\bullet, \bullet)$ Convolution Filter

Superscript or Subscript

$\bullet_{\infty}$ Free Stream Properties

$\bullet_{ref}$ Reference Parameter

$\bar{\bullet}$ Filtered portion of Variable

$\tilde{\bullet}$ Favre Filtered Portion of Variable

$\bullet^{u}$ Upwind Approximation

$\bullet^{c}$ Central Approximation

$\bullet'$ Sub-Grid part of Variable

$\bullet''$ Sub-Grid part of Variable in Favre Filter

$\bullet_{ij}$ Value at the Interface

Chapter 1

Introduction

This chapter presents the motivation and objectives of the current research. The chapter starts of an introduction of the significance of Low Reynold number flows ($Re < 10^5$) for VAWT applications. It follows with some general information about Mars' properties. Thereafter wind turbines and especially Vertical Axis Wind Turbines (VAWTs) are explained. The introduction to the physics of flow around VAWTs follows which includes the description of the differences between turbulence models. Consequently, we list the objectives of the current dissertation, and finally the outline of thesis is presented.

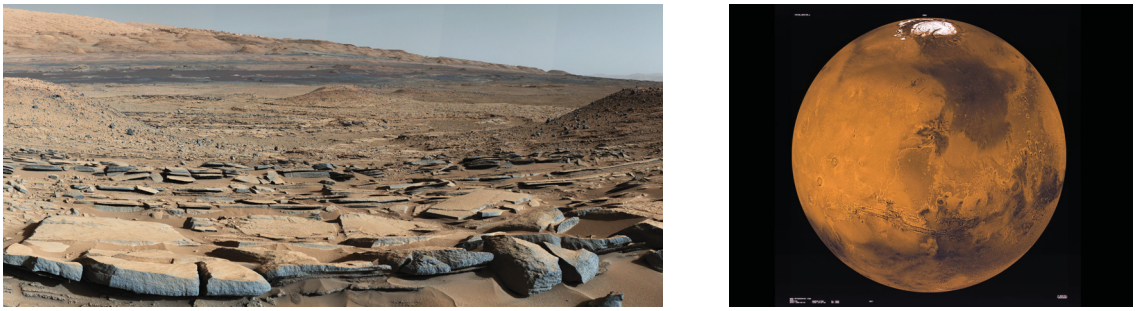
1.1 Low Reynolds Number Flows

The importance of flows in the range of Reynolds Number between 10000 to 100000 significantly increases in the last two decades. Re number, $(\rho UL/\mu)$, is directly proportional with the characteristic length of the simulated object and density of the free-stream. Therefore, low Re flows are frequently seen todays for smaller size vehicles and devices, and for missions to explore other planets with less density in comparison to Earth's. These applications include small Unmanned Aerial Vehicle (UAV) [16], and small size wind turbines that operate in urban areas [98]. In the current research, we concentrate on the simulation of a wind turbine on Mars. This is motivated by current plans proposed by aeronautics and aerospace companies to send human to Mars in the next

two decades, which investigation of devices to generate power using the available sustainable resources on Mars. Fortunately, low Re number enable us to utilize more accurate turbulence models such as wall-resolved Large Eddy Simulation, which are very expensive for high Reynolds flows.

1.2 Comparison of Mars' and Earth Atmosphere

Mars, fourth closest planet to the Sun in the solar system and the closest to Earth, is named after a Roman God, probably because of its reddish appearance (Figure 1.1). It is also commonly refereed to as the red planet. Mars' diameter is almost half of Earth's (Figure 1.2).



(a) **(b)**

Figure 1.1: (a) A view from the "Kimberley" formation on Mars taken by NASA's Curiosity rover.; (b) Viking 1 images composite of Mars by USGS University of Arizona. [39]

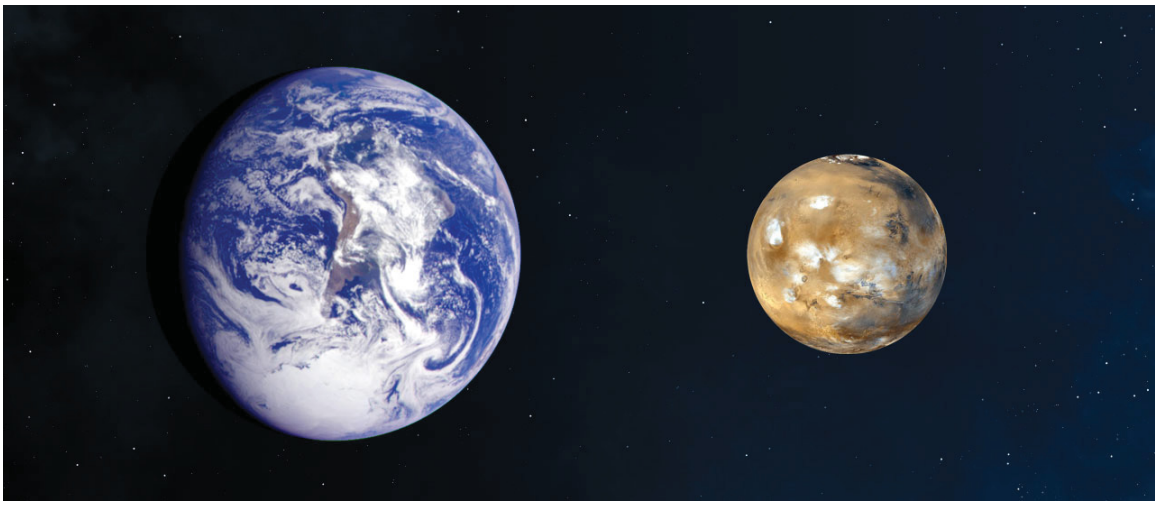


Figure 1.2: Size comparison of earth and Mars [75]

Due to similar tilts of the rotational axes of Mars and Earth, they experience similar seasonal patterns. However, a year lasts twice as long as the year on Earth due as it is further from the Sun (figure 1.3). Temperature changes from -143 $^{\circ}C$ in winter to 35 $^{\circ}C$ in summer. Similar to Earth, maximum and minimum temperatures happen at equatorial and polar caps, respectively. The length of a day is 24.6597 (hrs) which is very close to Earth's. More of Mars' properties are listed in Table 1.1. Due to higher kinematic viscosity on Mars for a similar characteristic length, L, and free stream velocity, U, Reynold Number on Mars would be less that 100 times of Earth's ($Re = UL/\nu$).

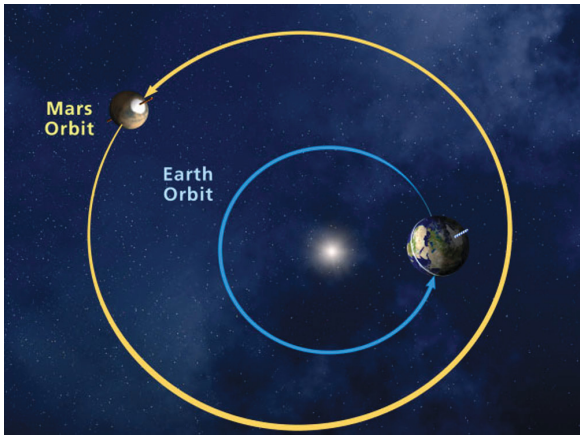


Figure 1.3: Orbit comparison of Earth and Mars [76]

Table 1.1: Comparison of Mars' and Earth's atmospheric properties.

Parameters	Mars	Earth	Unit
Surface gravity	3.72	9.81	$m.s^{-2}$
Speed of sound	229	321	$m.s^{-1}$
Mean molecular weight	43.4	29.0	$g.mol^{-1}$
Mean wind velocity	20	5	$m.s^{-1}$
Kinematic Viscosity	1.00E-3	1.5E-5	$m^2.s^{-1}$
Standard atmospheric density	0.015	1.225	$kg.m^{-3}$
Standard atmospheric pressure	700	101325	pa

The red planet experiences the largest dust storms in the solar system. Storms vary in size from regional storms, similar to dust storms on Earth, to continental-size storms. Almost every five years, these gigantic storms build up a global storm that covers almost the entire planet. Bigger storms can last weeks to months. The radioactive sunlight is the reason behind the dust storms on Mars.

(a) **(b)**

Figure 1.4: (a) A towering dust devil casts a serpentine shadow over the Martian. [76]; (b) A martian north polar dust storm. [77]

Heat transfers from the hot surface to its surrounding air. The warmer air then rises up carrying with it dust particles. In higher altitude warm and cool air together create an unstable environmental phenomenon such as a storm. During summer days, more radiative fluxes reach Mars' surface, therefore storms are observed more often during the summer, specially in the southern hemisphere.

Mars' atmosphere is almost 100 times thinner than earth's and it is more than 95 percent carbon dioxide. The breakdown of its composition is presented in Table 1.2.

Table 1.2: Breakdown of Mars' atmosphere composition and comparison with Earth's. [84]

Gas	**Abundance on Mars** $(\%)$	**Abundance on Earth** $(\%)$
Carbon dioxide (CO_2)	95.32	0.0397
Nitrogen (N_2)	2.7	78.084
Argon (Ar)	1.6	0.9340
Oxygen (O_2)	0.13	20.946
Carbon monoxide (CO)	0.08	—
Water vapour (H_2O)	0.03	0.25

Unlike Earth, we cannot generate most of our power from burning fuels because oxygen resources on Mars are very limited in compare to earth (Table 1.2). Then we need to look for alternatives energy production.

Radioactive and nuclear battery cells, such as Radioisotope Thermoelectric Generator (RTG), are reasonable options, and they have already been used and tested in some of current mission, such as the Curiosity rover, and previous missions (Viking). However, these devices are dependent on what

is built on Earth and then are transported to Mars by spacecrafts. In addition, it would not be wise to send human to another planet and leave them with only one source of power.

Harvesting natural energy in Mars is another alternative. Solar panel can be a good choice. However, previous missions have proved that we may not rely on them as a continuous power generator due to the frequent dust storms. At this point wind may help scientists on Mars. Wind blows almost constantly on Mars surface with the mean velocity of 20 (m/s), which is almost 4 times of Earth's. However, the density on Mars is less than Earth's (look at Table 1.1). Note that power is proportional to the cube of the velocity and only directly with density. Therefore, for a suitable location on Mars, with a high annual average velocity, a good energy production can be achieved.

1.3 Vertical Axis Wind Turbine

In the previous section wind energy was introduced as an available resource on Mars' surface. This can be used to generate power and make a habitable and sustainable environment for Mars' resident in the future.

In order to choose which type of wind turbines are more appropriate to operate on Mars, we need to consider few facts. Walking and travelling on Mars is far more difficult than on earth. In addition, exposing an astronauts to open areas for a long period of time put their life at risk. Thus, minimization installation and maintenance time of devices is crucial on Mars. In addition, choosing the most effective small size turbine $(< 100(kW))$ is vital.

Wind turbines are categorized as horizontal and vertical based on their rotational axis. The Horizontal Axis Wind Turbines (HAWTs) has a rotational axis along the wind direction, while it is perpendicular to wind direction in the case of Vertical Axis Wind Turbines (VAWTs) (Figure 1.5).

In general, HAWT's are more efficient than their opponents. However, there are some disadvantages that make them a less ideal choice to operate on Mars. Their generator, and gearbox are located on top of a tower. Then in order to perform maintenance, it is required to climb up to the top with a heavy suit and carry devices and repair them in a very harsh environment. In contrary, vertical axis wind turbines' generator and gearbox are located close to the ground that makes them easy to access and maintain. In addition, it also means that installation of a Vertical axis wind turbine needs

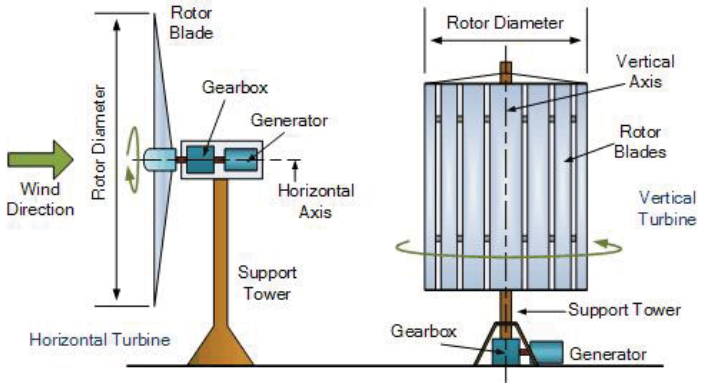


Figure 1.5: Horizontal (left) and Vertical Axis wind turbine (right) [117]

less time and effort. Another problem with horizontal axis wind turbine is related to wakes that they created. HAWTs create larger wakes behind them in comparison with VAWTs. This increases the clearance distance between wind turbines. On the other hand VAWTs can be packed closer together in wind farms due to their smaller wakes. Before we start talking about advantage of VAWTs, two variable, that are frequently used in wind turbine industry, are defined.

Tip Speed Ratio (TSR) is ratio of tip velocity of turbine's blade to wind speed,

$$\lambda = \frac{R\omega}{U_\infty} \tag{1.1}$$

where R is rotor radius, ω is angular velocity at tip of blade, and U_∞ is the wind speed. Another important parameter is the power coefficient which represents harnessed power by turbine to the available kinetic wind energy.

$$C_p = \frac{P}{0.5\rho_\infty U_\infty^3 A_s} = \frac{T.\omega}{0.5\rho_\infty U_\infty^3 A_s} \tag{1.2}$$

where P is the power, T is torque imposed on rotor from wind, and ρ_∞ is free stream density. A_s is swept area by rotor. According to the Betz's law [11] C_p cannot exceed more than 59.3% for any type of wind turbine.

Figure 1.6 compares average wind turbine power coefficient versus tip speed ratio. HAWTs can generate more power, however they need to operate at higher TSR. Higher tip speed ratio means we need larger rotors and also they need to be mounted higher in order to avoid effect of the ground's

boundary layer. This is the main reason that there is more demand for VAWTs in urban area where space availability is limited and average wind velocity is less.

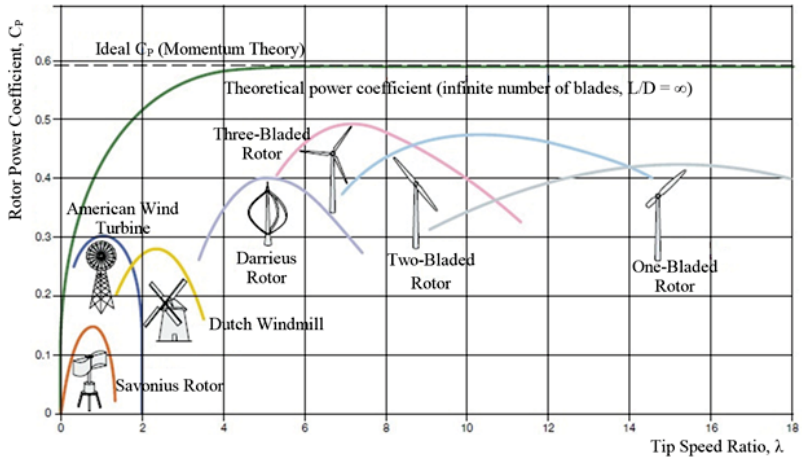


Figure 1.6: Power coefficients of wind rotors of different designs [29]

Aforementioned characteristics of VAWTs make them a better choice for operating on Mars. Now, we need to choose which type VAWT is a the best choice depends on their performance.

(a) **(b)**

Figure 1.7: (a) Savonius; (b) Darrieus. [1]

Vertical axis wind turbines are categorized as Savonius and Darreius type (Figure 1.7). Savonius turbines are rotated by the drag component of force on their blade, while lift is the torque producing force on a Darrieus turbine. Self-starting ability of Savonius type is the most important advantage

over the Darrieus type. However, Savonius turbines usually generate less power than Darrieus ones, as it is seen in Figure 1.6. Thus, Darrieus type can reach a comparable power coefficient to the HAWT's type with a few modification. That is the reason we concentrate on Darrieus turbine in this dissertation.

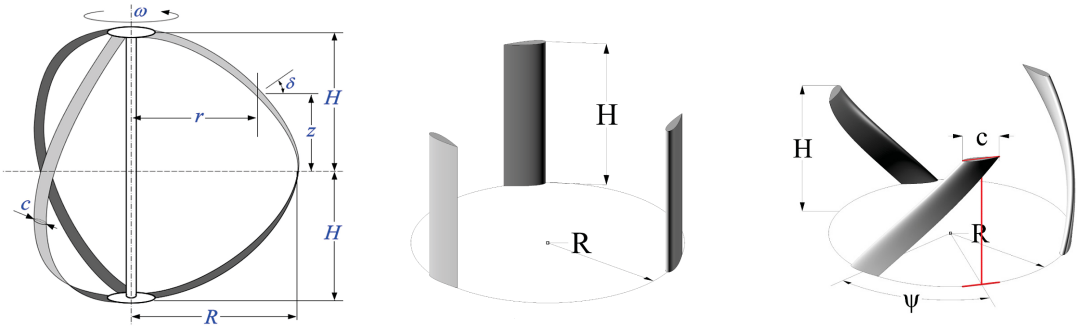


Figure 1.8: Three types of Vertical Axis Wind Turbine; (a) Darrieus blade [108].; (b) H-Blade [2].; (c) Helical balde [2].

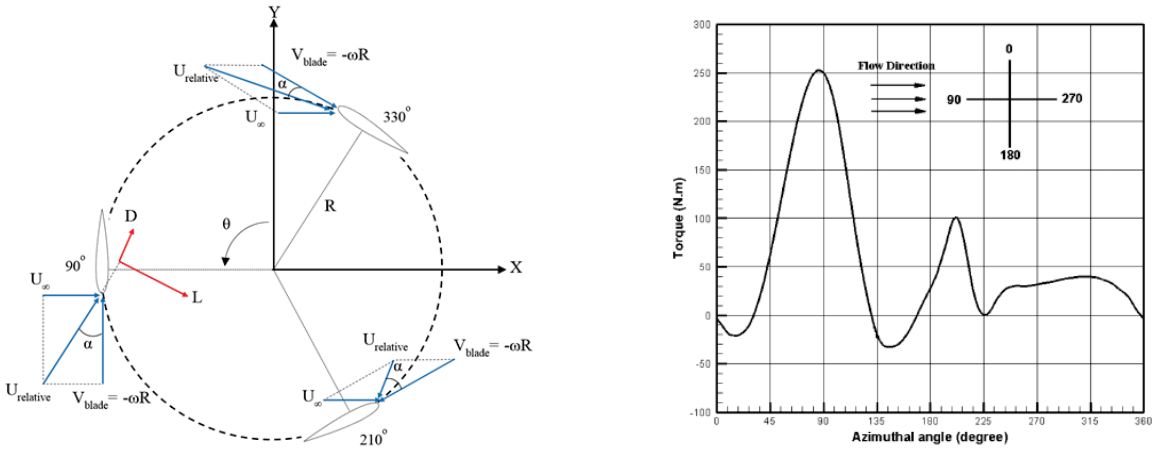
The Darrieus turbine can be designed in different shape. Three of them are shown in Figure 1.8 H-type turbine was one of Darreius' patent in 1927. In spite of its simple geometry, flow complexity make them one of the most challenging problem in aerodynamics.

In order to improve turbines' performance, one needs to reach an accurate comprehension of their aerodynamics. Study of flow around VAWTs still is less mature in comparison to HAWTs'. Similarity of aerodynamic behaviour of HAWTs to helicopters' blade is an undeniable factor that has helped to optimize their design during almost half a century. Lots of resources have flowed into this research area because of military interests in helicopters, and similar to a good deal of other aerospace and mechanical engineering project, after maturing in the military for a long time, some civil usages also rose from them.

1.3.1 Aerodynamics features of vertical axis wind turbines

Different strategies have been developed to increase the efficiency of VAWTs. But, before we tackle performance improvement, we need to review the aerodynamics of VAWTs. Figure 1.9

shows Lift and Drag component of the force on a turbines' blade due to pressure difference and shear stresses. It also demonstrates how loads on blades typically change over a cycle.



(a) **(b)**

Figure 1.9: (a) Forces on blades of H-type at different azimuthal angle θ [40]. (b) Typical torque variation over a cycle [88]

Herein we summarized a few aerodynamic features of H-blade vertical axis wind turbines that have been observed and studied. Dynamic stall, vortex shedding and interaction of wake and vortices with blades are commonly observed in VAWTs simulations.

Although, at a constant tip speed ratio, both incoming velocity and angular velocity remained constant, direction of blade's velocity changes constantly. Consequently, value and direction of relative velocity of wind with respect to the blade vary. For a symmetric airfoil maximum torque occurs at or around 90 degree, and a second pick at 270 degrees. At lower tip speed ratio AoA's variation is more severe. The equation follow as,

$$\alpha = atan(\frac{\sin\theta}{\cos\theta + \lambda}) \tag{1.3}$$

where θ is the azimuthal angle that varies from zero to 360°. This is shown in Figure 1.10 for TSRs between 1 to 5. It is seen that for low values of λ, the angle of attack is mostly above 10°.

In an steady simulation by increasing the angle of attack, the lift coefficient also increases up to the point that it starts falling which is called static stall angle (α_{ss}). In the static mode usually tailing edge stall happens for thick airfoil profiles, while leading edge stall is more common for

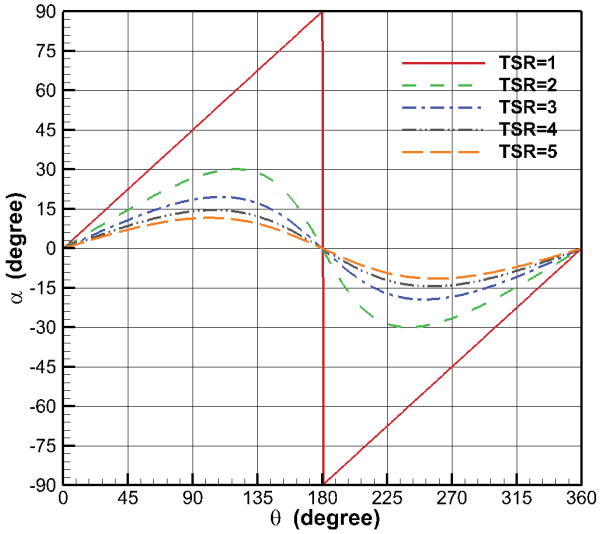
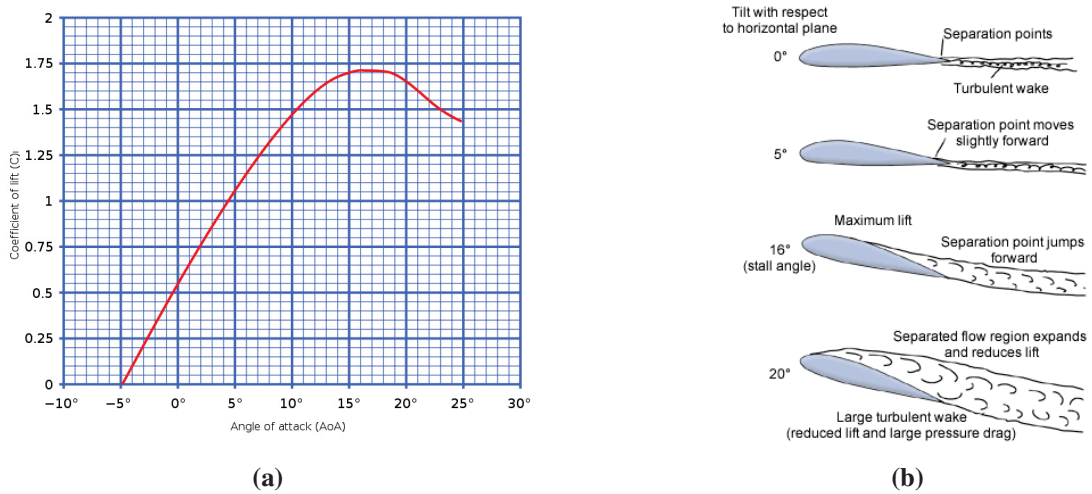


Figure 1.10: Angle of attack variation of a H-type VAWT's blade at different tip speed ratio

thin airfoils. An example of lift coefficient curve is demonstrated in Figure 1.11a for a trailing edge stall. As angle of attack increases flow accelerates on the suction side of blade. Flow eventually is separated very close to the trailing edge and a narrow wake generates just behind the airfoil. At stall, separated flow moves toward the leading edge, and a completely turbulent flow cover most of surface on the suction side. Beyond the stall point a passage of vortex start to detach from surface and travels downstream. The septation starts very close to the leading edge, and then it extends to downstream. Usually a sharper drops is observed at leading stall in comparison with the trailing edge stall.

Dynamic stall only occurs on a moving blades due to rapid changes of AoA. Leading edge stall are more common on oscillating blades. Figure 1.12 compares lift curve between static and dynamic lift. It is seen that for lower values than α_{ss}, the curves are very close, after the static stall point a small vortex appears close to the the leading edge. That exceeds C_l from stall. This vortex grows and moves toward the trailing edge. After it separate from airfoil trailing edge a sudden and severe drops happens to the lift coefficient. Then maximum lift is higher in dynamic stall and it happens at larger angle of attack than α_{ss}. However, a sharp drop causes to loss most of the lift.



(a) **(b)**

Figure 1.11: (a) Lift coefficient versus AoA for a static airfoil [15]. (b) Flow pattern around airfoil [101].

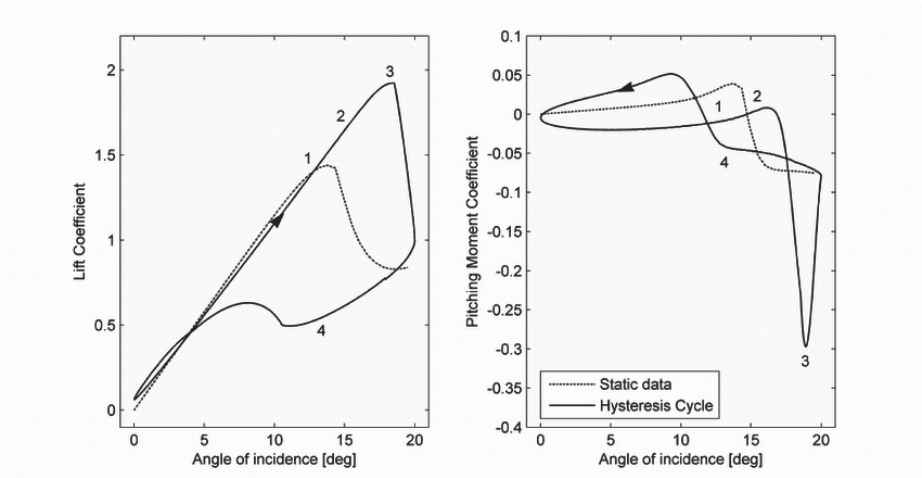


Figure 1.12: Comparison of static and dynamic stall [64]

McCroskey [66] explained the phenomena of dynamic stall for an oscillating blade. He performed a comprehensive study of different regimes of dynamic stall (light and deep stall) and investigated many geometrical and flow parameters such as blade profiles, Mach number, frequency and mean angle of attack on formation of dynamic stall. McCroskey [67] published another paper in the same year on effects of airfoil profile on dynamic stall. Laneville and Vittecoq [56] investigated formation of dynamics VAWT at different tip speed ratio. He studied Reynolds number $Re = 3.8 \times 10^4$ and concluded that for the NACA0018 dynamics stall expected from TSR less than 4 where a light dynamic stall happening, while a deep stall occurs at lower TSRs $(TSR < 2.5)$.

1.3.2 How to improve performance of H-type Vertical Axis Wind Turbines

In order to improve performance of small vertical axis wind turbines ($< 100KW$) two major methods are employed:

(1) Accelerate velocity just upstream of turbines;

(2) Modify angle of attack of turbines' blades.

In the first method, the objective is to accelerate flows as they approaches rotors. From the fluid mechanics and the continuity equation we know that inside a diffuser, a subsonic flow decelerates as the area increases. Therefore, a wind turbine is placed close to the entrance of a diffuser. Usually diffuser are accompanied by a flange-shaped surface at the end of diffuser. Flange creates a low-pressure zone in downstream of turbine that draws more flow into the turbine (Figure 1.13). Since the Betz's laws and power coefficient are derived based on an undisturbed uniform flow, there is a possibility that wind turbine power coefficient can even exceed the Betz's limit.

Diffuser-agumented wind turbine have been built and tested for small horizontal wind turbine [82]. They showed that the power coefficient of turbine exceeds Betz's limit. Krishnan and Paraschivoiou [55] conducted a comprehensive CFD research on flanged diffuser for wind turbine system. They improve the performance of the basis design. Geurts et.al [38] and Watanabe et. al [116] independently improved the performance of H-type VAWT. It is noteworthy to mention that placing a wind turbine inside a cascade introduce some negative features as well. This includes blockage, leaking, and secondary flow losses.

Other way to increase the efficiency of VAWT is to manipulate the angle of attack such as maximize power over a cycle. Therefore a mechanism is required to change the AoA of blade. This either can be performed by adding some mechanical devices that spin the blades locally, or use flap and slap similar to airplans' wing, or we can use smart material to deform the blade. Herein, a few papers in the literature that have investigated the effect of shape of blades and their pitching angle on the performance of wind turbines are summarized.

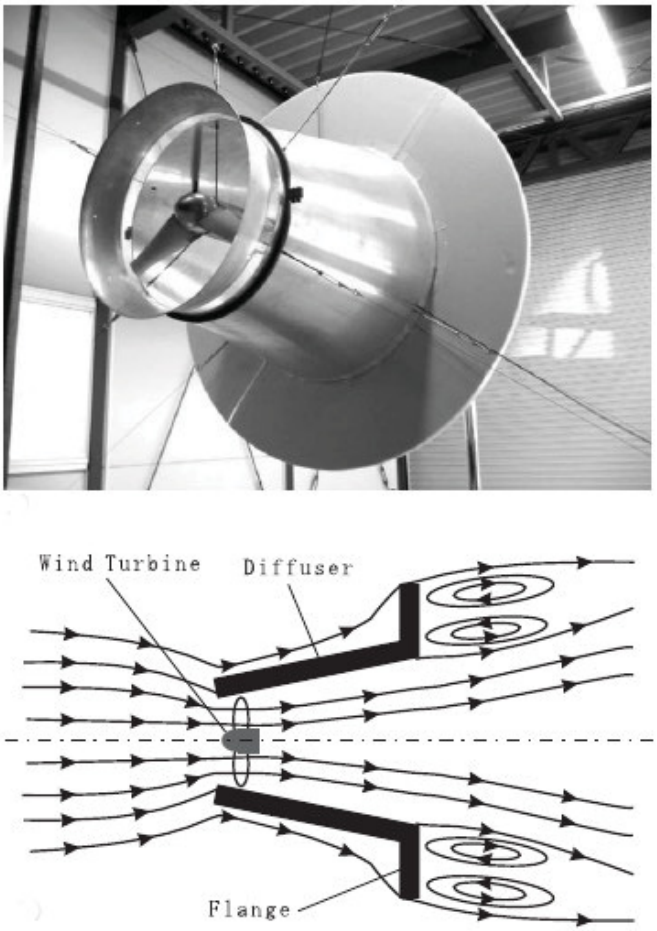


Figure 1.13: (a) diffuser-shrouded wind turbine; (b) flow mechanism around a flanged diffuser. [82]

Mohamed [71] conducted a 2-D study with the Realizable $k-\epsilon$ model on medium size grids. He investigated the effects of 20 airfoil profiles on the maximum power coefficient over tips speed ratio between 2 to 20. His results shows an interesting fact that using high-lift airfoil and specially non-symmetric airfoils does not results in a better power output necessarily. In most of the cases, symmetric or close to symmetric airfoils demonstrated a higher efficiency. He concluded that the symmetric airfoils delay dynamic stall and therefore they generate more power over a cycle. Mohamed et al. [71] questioned his conclusion in his follow up paper. He concluded that the $k-\omega\ SST$ with a finer mesh close to airfoils results in more reliable simulations. He simulated 25 airfoil's profile shapes this time, and the paper showed that a non-symmetric airfoil increases the efficiency of the best symmetric airfoil (NACA0018) by 9%. He also conducted the pitching angle study for one

of non-symmetric profile with the highest power coefficient. Zero pitching angle showed the highest performance between all range of pitching angle from -10 to 10.

Bausas et al. [8] reached to a similar conclusion as Mohamed. They compared a symmetric airfoil (NACA0025) with a 1.5 percent cambered (NACA1425) airfoil under a steady inlet and unsteady flow conditions. Both conditions showed that non-symmetric airfoil generate more power (at least 11%) in compare to the symmetric one. They used $k-\omega\ SST$ model with a fine grid ($y^+ < 1$). Note that the turbulence model, that used in Bausas's study, is more accurate than Mohamed's work.

Xiao et al. [123] simulate a vertical axis tidal turbine (VATT) with three blades using Realizable $k-\epsilon$. They compared power coefficient between three configurations: a) NACA0018 b) slotted blade with a fixed flap c) slotted blade with an oscillating flap. Their results showed that using two components airfoil delays the dynamic stall due to moving the separation point to the trailing edge. In addition, they proved that choosing an appropriate oscillating amplitude and frequency reduces the blade and wake vortex interaction. They could increase power of the VAWT by 28 percent.

Miau et al. [70] suggested a variable pitching system to control the angle of attack at different blade's location. They studied a H-blade (NACA0018) vertical axis wind turbine both experimentally and numerically with using $k-\omega\ SST$ method using ANSYS Fluent. They concluded that a wind turbine with a variable pitching angle outperforms one with a constant pitching angle. Miau et al. Claimed that a sufficiently high pitching angle may solve the self-starting issue of wind turbine.

Chen et al. [20] conducted a 2-D CFD study (100,000 cells) to analyze selt-starting ability and performance of H-type VAWT with three blades using FLUENT. They used $k-\omega\ SST$ as turbulence model. Their results showed that cambering an airfoil positively may alleviate its self-starting problems. However, the maximum camber does not necessarily result in the highest power output. Their results also showed that the blade with the largest camber is less sensitive to pitching angle between -10 to 10 degrees.

Wolff et al. [121] performed a 2-D study (40,000 grid points) of a morphing blade. They used

Spalart-Allmaras model with $(y^+ < 1)$ to simulate a single blade that goes under deformation. They especially focused on reduced frequencies. Danao [24] studied the effect of camber and thickness of blade with various URANS models. Their research include 6 different blade profiles, and they concluded that $k - \omega\ SST$ predicts the closes results to the experimental simulations. They concluded that a thinner symmetric airfoil provides the highest overall power. However, by sightly cambering the airfoil we can improve the performance in some portion of cycle both in downstream and upstream. They also visualize the differences of the vortex formation and detachment from the surface of the blade at different location. They showed the dynamic stall and vortex shedding completely changes from one blade to another.

Recently, there also have been few articles that used LES to study small H-type VAWTs. For example, Kanner et al. [49] performed both 2-D and 3-D CFD Implicit Large Eddy simulation (ILES). Their results showed that for pre-stall angles both for 2-D and 3-D models deliver very accurate simulation. However, post-stall angles only 3-dimensional model match the experimental simulation. Li et al. [61] applied a 2.5 LES model to simulate an H-type VAWT . They showed that LES especially at low tip speed ratios predict the most accurate results compared to URANS models.

1.3.3 Tools to model flows around VAWTs

The study of VAWTs can be categorized into three main fields [48]: 1) Experimental Study 2) Aerodynamic Models 3) Computational Fluid Dynamics (CFD) models (Figure 1.14).

Aerodynamics Models

Aerodynamics models use numerical models that are bases on Aerodynamic and Fluid mechanics background. One of widely used in this field is momentum model. Momentum models initially were based on considering a single streamtube [109]. They calculated the aerodynamic forces of wind turbines by using the momentum equation. The single streamtube model is very fast and it gives a good sense about the overall performance of wind turbines. Though usually it over-predicts the power, and cannot provide information about velocity distribution around rotors.

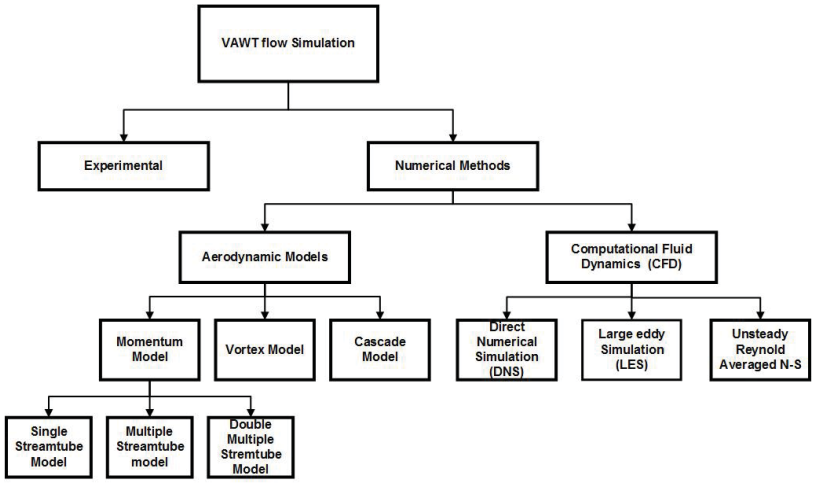


Figure 1.14: Chart of method to simulation VAWT

In 1974, Wilson and Lissaman [120] introduced the multiple streamtube model in order to improve some drawbacks of single streamtube. In spite of many efforts and modification to improve multiple streamtube model [46, 74], the model still under-predicts power coefficients. Lapin [57] introduced a new method that used double actuator discs instead of one actuator disk. Paraschivoiu [87] elaborated Lapin's concept and combined it with new idea of double-multiple streamtube model (DSMT). It has proven that DSMT outperforms the rest of momentum models. Still, it may results in wrong answer at high angle of attacks.

Vortex model originated from potential flow that has been used in the avionic industry [44, 58].However, it works only for small AoA. Cascade models that were proposed by Hirsch and Mandal [43] to simulate VAWTs at high tip speed ratios.

Computational Fluid Dynamics Simulations

Although Aerodynamics models have proven tools, they lack to provide accurate information about flow features that happens near the blades' surface. Computational fluid dynamics can fulfil the gap. In CFD models, Navier-Stokes equations, that define the flow properties of fluid flows, need to be solved. Therefore, CFD methods are based on discretization of domain and fluid mechanics governing equations. They are usually more expensive than Aerodynamics models, but they provide a comprehensive data from the far-field to the surface of blades. They are also valuable tools to

understand the physics of a flow in details.

For unsteady turbulent fluid dynamics, such as the flow related to VAWTs' simulation, can be model with Direct Numerical Simulation (DNS), Large Eddy simulation (LES), or Unsteady Reynolds Averaged Navier-stokes (URANS) equations. Figure 1.15 shows the cascade of turbulent kinetic energy in a turbulent flow. Instability is introduced to the flow due to small perturbation inside the mean flow. Then, some organized turbulence features (eddies) are build up from the mean flow. These large eddies contain the majority of turbulent kinetic energy of flow. However, they are not stable and they break-down to smaller eddies. The process continues, inside the so-called inertial subrange, down to Kolmogorov length scale ($\eta = (\nu^3/\epsilon)^{1/4}$) where the molecular viscosity of flow becomes dominant and the kinetic energy dissipates into the heat. The kinematic viscosity, ν, of the fluid and the dissipation rate, ϵ are quantities that defined this flow.

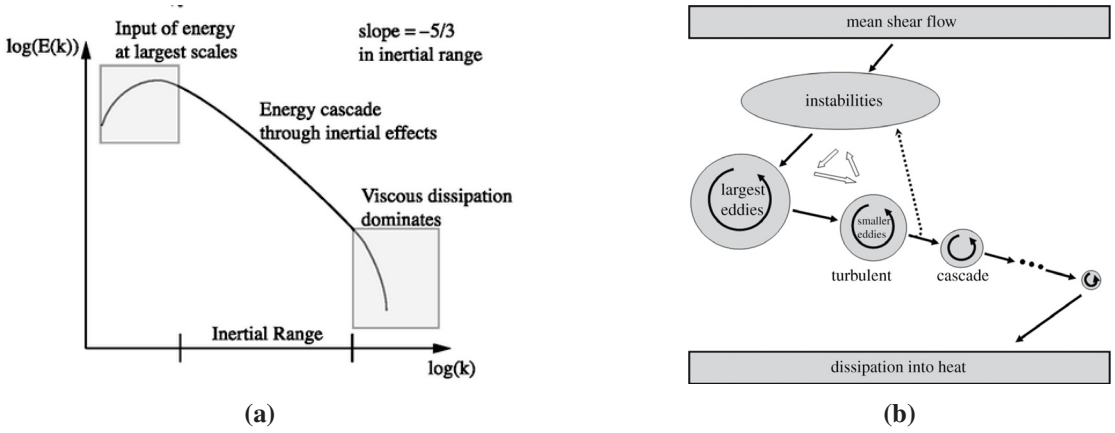


Figure 1.15: (a) Cascade of kinetic energy [10].; (b) Direction of turbulence scale from mean flow to Kolmogorov scale [51].

In order to simulate a flow, one needs to predict this cascade accurately. Differences between CFD models related to how they predict the energy cascade. As it is seen in Figure 1.16.

Direct Numerical Simualtion

DNS resolves all the eddy's scales down to the Kolmogorov length scale ($Re^{-3/4}$). This also dictates the minimum grid size required in computational fluid dynamics. Turbulent structures are inherently 3-dimensional features. Thus, a 3-dimensional grid requires more than $Re^{9/4}$ cells to resolve 3-D coherent structures. It means that to simulate a flow at Reynolds number $O(10^5)$,

which is the Reynolds number of most of VAWTs, DNS requires $O(10^{10})$ grid nodes. Even with the recent improvement both in software and hardware of super computers, it is almost impossible to use DNS for a real engineering problem. Computational cost is not the only barrier that DNS model is facing to become an affordable model in the near feature. Implementation of initial and boundary conditions are critical when a DNS model is employed. Since all the turbulence scales are resolve, any non-physical disturbance in the inlet or initial conditions may change the flow pattern completely. It is also very important that truncation errors that emerges from the numerical schemes do not produce any extra disturbance to the flow.

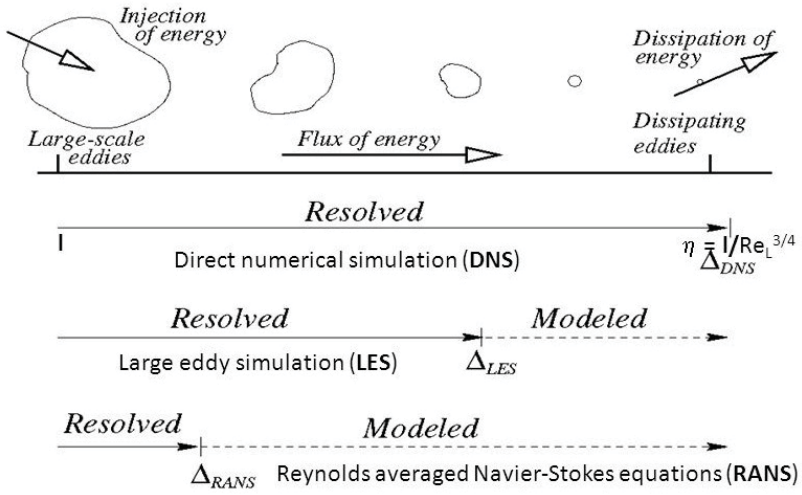


Figure 1.16: Comparison of DNS, LES, and URANS model to capture the cascade [45].

Unsteady Reynolds-Averaged Navier Stokes

URANS models are on the other side of spectrum. Instead of resolving the inertial subrange, they are completely modeled. Models are based on mathematical manipulations, dimensional analysis and some empirical constant to calibrate the equations [119]. Their objective is to predict some turbulence parameters such as kinetic energy, and dissipation energy. In the past three decades these methods have been developed and used in a wide range of engineering applications. These models can be lowered on an algebraic equation [6, 100] such as mixing length model to more complex model such as Reynolds stress model by adding seven extra equations to the Navier-Stokes equations. Spalart-Allmaras [103] and $k-\omega$ SST [69] are the most popular ones for VAWT simulations

. However, there are some concerns about using them for an unsteady turbulent flows.

Some raise from inherent disadvantages of RANS models. These models are based on the statistical average in time, even for an unsteady simulation. Therefore, in the case of inherently dynamic flows, they fail to capture some of turbulent features [10]. In addition, turbulent structures are completely 3-dimensional phenomena, while most of the study that performed with URANS models are based on 2-dimensional simulations.

Large Eddy Simulation

Large Eddy Simulation, introduced by Smagorinsky [99], uses some mixed of both DNS and RANS model. Instead of modeling the entire scale of energy cascade, those that fall under a certain wave length are modeled and the rest are resolved. Therefore a space filter is employed instead of time averaged, used in the RANS model. Wave length that separates resolved eddies from modeled ones is called cut-off wave length. Smaller eddies, especially those near to the Kolmogorov length scale, can be considered homogeneous and universal. Therefore, their model are simpler than RANS models. These models are so-called sub-grid scale (SGS) models. Subgrid scale models named owing to proportional of the filter size to the grid size in LES methods. Some authors [89, 91] suggest that sub-filter scale (SFS) is a more appropriate name for small-eddy's models.

LES of free shear flow requires considerable less number of grid nodes (Re) in comparison to the DNS model [89, 91], since the majority of inertial transfer happen at large scales. In contrary, very high resolution is required to simulate a complete 3-dimensional flow with wall-resolved LES models. That is because transfer of energy from larger eddies to smaller ones continues up to the Kolmogorov length scale inside near-wall regions, and they play a very significant role in the simulation of turbulent features.

Chapman [19] used the friction coefficient and the seven-power velocity distribution law to estimate required number of grid points for a Large eddy simulation. He conducted his study based on two scenarios: 1) Wall-model LES, which the inner-layer zone of boundary layer, that include between 10 to 20 percent of the boundary layer, are modeled with log-law. 2) Wall-resolved LES

model, where the inner-layer of boundary layer are resolved with LES. According to Chapman calculation number of grid points for a wall-modeled LES is proportional to $Re^{0.4}$, while it significantly increases to $Re^{1.8}$ for wall-resolved LES simulation. Chapman evaluated his calculations based on friction coefficient for a low to intermediate Reynolds ($Re < 10^6$). Choi and Moin [21] revisited Chapman's criteria of grid requirement for flows at high Reynolds number. They suggested that the grid size required for a 3-D wall-resolved LES simulation is proportional to $Re^{(13/7)}$), while wall-model LES requires (Re) grid points.

Both Moin's and Chapman's estimations show that a very fine grid is required for a wall-resolved LES simulation at high Reynolds. Therefore, currently the majority of LES simulations at high Reynolds number ($Re > 10^6$) use wall-model LES [17, 78]. For wall-modeled LES, a log-law is implemented to calculate the shear stress on the wall from the information that are provided from the resolved LES of the rest of domain, then calculated shear stress is used as the boundary condition for LES. Therefore, first grid off the wall is situated inside log-layer which results in lees expensive mesh near-wall region. Although, wall-model LES considerably reduces the computational cost, especially at high Reynolds number, the accuracy of the method depends on accuracy of information that transfer from wall model to LES and vice-versa. Usually Wall-model LES works well for attached flow, however its accuracy drops for separated flows. These models also result in more accurate solution of flows at high Re in compare to low Re ones. Moreover, these models can not be used to model the transition from laminar to turbulent flows without adding extra equations.

Another solution for remedy of high cost of wall-resolved LES simulation is to use hybrid LES/RANS models. These methods use RANS model in near-wall regions and LES elsewhere. Spalart [102] was one of pioneer who first introduced Detached Eddy Simulation (DES). It combines LES with Spalart-Allmaras model. DES uses Spalart-Allmaras inside the attached boundary layer region, and LES is used in the separated zone. Strelets [105] combined $k - \omega$ Shear Stress Transport (SST) with LES and investigate transitional flow from laminar to turbulent flow. Although DES proved that it improves the time-variation of eddies around the surface and requires considerable smaller number of grid points in comparison to regular LES, prediction of separation point mainly depends on the accuracy of RANS model. Nikitin et.al [80] showed that DES simulation

usually underpredict the wall shear-stress.

Fortunately with the recent improvement of in hardware technology and grows of supercomputers in the field of CFD, wall-resolved LES is affordable for low Re flows. Wall-resolved LES with the sufficient grid residual, and acuurate model and numerical scheme can results in more accurate solution in compare with both hybrid LES/RANS models and wall-modeled LES. Number of grid nodes also for LES simulation can reduce for 2.5-D simulations. 2.5-D models refers to simulation of 2-D flows, for example flow around airfoils, with inherently 3-D turbulence models. Therefore, there is no corner and secondary flow to be captured, that reduces the grid size. Grid is extended in the normal direction for 2.5-D models large enough to capture the forming and bursting of the eddies close to wall surfaces, and usually a periodic boundary condition is applied in the normal direction.

The standard eddy viscosity model [99] assumes an equilibrium between the dissipation and the production of energy at the small scale level, where flow features are modeled. With a sufficient fine grid and an appropriate Smagorinsky constant $(C_s = 0.1 - 0.2)$, the standard model accurately predicts flow for isotropic flow at high Reynolds number. However, some difficulties emerges for wall-bounded flows. The standard Smagorinsky does not predict the asymptotic behavior of eddy viscosity near the wall. In addition, value of eddy viscosity does not go to zero at wall. In order to ensure the asymptotic behavior of eddy viscosity and shear stresses near-wall region a damping function has been suggested by some authors [72, 90]. Germano et.al [37] introduced more robust method of dynamic model based on Smagorinky method. Dynamic models use two filter levels instead of one. First one is called test filter, and it uses to calculate the Smagorinsky constant locally. Then, the second filter calculated the shear stress. Germano et.al showed that a dynamic model resolve the well-known problems with the standard Smagorinky model, i.e. asymptotic behavior, simulation of transition to turbulence. The dynamic model is very common for structured mesh, it also uses with unstructured mesh. Another drawback of the standard eddy viscosity is related to that fact that, it only predict eddy viscosity for the region that strain rate is not zero. In real case there are some region of flows that vorticity are dominate, so-called Eddy regions. Inside eddy

region, although there stain rate can be zero, the eddy viscosity is non-zero. Nicoud and Ducros [79] suggested a new model based on both vorticity and strain rate, Wall Adapting Local-Eddy viscosity (WALE). They showed that their model not only improve the accuracy of simulation inside the eddy region, it also predict the asymptotic behavior of eddy viscosity near the wall. Moreover WALE is calculated the eddy viscosity locally with an algebraic equation. Therefore, it is less expensive that dynamic models, which requires solving the equations on two grid levels.

The researches that were listed in the previous section provide valuable information regarding to shape and pitching angle of blades. However, the majority of them are based on RANS models, and their accuracy is questionable, especially at lower TSRs. Therefore, our research is focused on 2.5-D simulation of VAWTs with a LES-based code. Considering the significance of capturing the physics around the blade surface for VAWT simulations, WALE is used in the current research to model small grid scale turbulence. In addition, we use a morphing trailing edge profile to control the pitching angle. This method has been developed by Prof. Inman and his team [85] at University of Michigan.

1.4 Objective

The objective of current research are summarized as follows,

- Introduce a new design of small wind turbine that operates on Mars surface based on VAWT with morphing blades
- Develop and validate an accurate WALE-LES based CFD code. This model should be able to predict 3-dimensional and 2.5 D turbulent flows with at least second order of accuracy. It is very important that numerical scheme owe to be a low-dissipative one.
- Reduce the computational cost of the in-house code by choosing the right parameters that gives the most accuarte solution in the shortest CPU time.
- Develop some tools to dynamically deform 3-dimensional grid to simulate a vertical axis wind turbine when morphing the blade.

– Study the physics of dynamic stall and vortex shedding by visualization at flow contours and analysis of variation of lift and drag on the blade.

– Perform LES simulations for VAWT with deforming blade.

1.5 Thesis Outline

Chapter 2 describes the governing equations of fluid flow for VAWTs. The auxiliary equations for a real gas, and implemented turbulence models are listed. Non-dimensional form of the governing equations that are used in the current project are also derived. Chapter 3 presents numerical methods that are employed. Details of the discretization technique in space (diffusion, convection, and source term) and in time are discussed. Thereafter, available boundary condition and their implementation are shown. Finally, a brief summary about the linear solver techniques and their efficiency is provided. We will introduce both mathematical and numerical procedures of the mesh deformation in chapter 4., where spring and Laplace methods are described in this manner. Consequently, we talk about the Arbitrary Lagrangian Eulerian (ALE) method and how it can grantee second accuracy of the numerical scheme. We allocate chapter 5 to demonstrate the convergence study and mesh deformation analysis. Finally, chapter 6 is allocated to use our in-house code to simulate a steady blade at high angle of attacks in order to validate WALE simulation for flows with large separation regions. It follows with three fixed-blade VAWT Simulations. These include a downward, upward, and close to symmetric profiles. Load and power generated with each blade are compared. Finally, capability of our developed code to simulate a morphing blade VAWT is presented. Chapter 7 concludes our dissertation.

Chapter 2

Mathematical Models

2.1 Introduction

To solve unsteady turbulent flows, the governing equations are first introduced. The Navier-stokes (N-S) equations are the most general governing equations to simulate the Newtonian fluid flows in the single phase state. These equations describe conservative of mass, momentum and energy. In addition, depends on type of problem, some auxiliary equations may be required to close the system of equations. For example state equation of an ideal gas or transport equation in multi-phase flows. The N-S equations are a system of Partial Differential Equations (PDE's). Analytical solutions of them are only available for very simple flows and geometries because of the complexity and non-linearity of the system.

Flows are categorized as laminar or turbulence. In the former, only molecular viscosity dominates the shear stress and diffusion of the flow. In contrast, the shear stress and diffusion terms are influenced by both molecular viscosity and local flow parameters in turbulent flows.

Direct Numerical Simulation (DNS), Large eddy simulation (LES), and Reynolds Averaged Navier-Stokes (RANS) equations are three main methods to solve turbulent fluid flows. In DNS, the Navier-Stokes equations are resolved without adding or elimination any term from the fluid parameters. While in Large Eddy Simulation a space-averaged quantity of the flow parameters has

to be calculated and the influence of smaller eddies are taken into account by adding a turbulence model. Finally, a time-averaged of the flow parameters are resolved and some extra equations or terms are supplied to model the turbulence parameters in RANS models.

In this chapter, first, the N-S equations and their closure equations are described. Second, the governing equations based on the space-filtered quantities are extracted. Finally, the Large Eddy Simulation model used in this research is represented.

2.2 Navier-Stokes Equations

Simulation of flows contains evaluation of the flow and thermodynamic properties at each point in space and time. For a compressible flow, unknown parameters usually include density, velocity, and pressure. Temperature or entropy sometimes are used instead of pressure. The conservative form of the Navier-Stokes equations is,

$$\frac{\partial \boldsymbol{W}}{\partial t} + \nabla.({}^{c}\boldsymbol{F} + {}^{v}\boldsymbol{F}) = \boldsymbol{S} \tag{2.1}$$

which $\boldsymbol{W}$ contains conservative variables; ${}^{c}\boldsymbol{F}$ and ${}^{v}\boldsymbol{F}$ include convective and viscous fluxes, respectively, and source term ,$\boldsymbol{F}$.

$$\boldsymbol{W} = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho E \end{bmatrix}, \quad {}^{c}\boldsymbol{F}_i = \begin{bmatrix} \rho u_i \\ \rho u_i u_1 + p\delta_{i1} \\ \rho u_i u_2 + p\delta_{i2} \\ \rho u_i u_3 + p\delta_{i3} \\ \rho u_i h \end{bmatrix}, \quad {}^{v}\boldsymbol{F}_i = \begin{bmatrix} 0 \\ -\sigma_{i1} \\ -\sigma_{i2} \\ -\sigma_{i3} \\ -u_1\sigma_{1i} - u_2\sigma_{2i} - u_3\sigma_{3i} + Q_i \end{bmatrix} \tag{2.2}$$

where ${}^{c}\boldsymbol{F}_i$ and ${}^{v}\boldsymbol{F}_i$, $(i = 1, \ldots, 3)$ are the conservative and diffusive fluxes vectors, and the total energy is E,

$$E = e_I + \frac{1}{2}u_i^2 \quad , i = 1, \ldots, 3 \tag{2.3}$$

where e_I is the internal energy and δ_{ij} is Kronecker delta,

$$\delta_{ij} = \begin{cases} 1 & , \quad i = j \\ 0 & , \quad i \neq j \end{cases} \tag{2.4}$$

where h in equation (2.2) is enthalpy,

$$h = e + \frac{p}{\rho} \tag{2.5}$$

where σ_{ij} are viscous stresses which are functions of the rate of deformation for a Newtonian fluid,

$$\sigma_{ij} = \mu(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}) - \frac{2}{3}\mu \; div \; \boldsymbol{u} \tag{2.6}$$

divergence of velocity is given,

$$div \; \boldsymbol{u} = \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_3}{\partial x_3}$$

where μ is dynamic viscosity and it varies with temperature according to Sutherland equation,

$$\mu = \mu_s(\frac{T}{T_s})^{\frac{3}{2}}(\frac{T_s + T_c}{T + T_c}) \tag{2.7}$$

the values of constant μ_s, T_s, T_c depend on gas types. For air and Mars' atmosphere are shown in Table 2.1,

Table 2.1: Sutherland constant

Gas	$\mu_s[kg/(m.s)]$	$T_s[K]$	$T_c[K]$
Air	1.711×10^{-5}	237.15	110.4
Mars' Atmosphere	1.480×10^{-5}	293.15	240

heat flux, Q_i , is defined by the Fourier law,

$$Q_i = -k\frac{\partial T}{\partial x_i} \tag{2.8}$$

where thermal conductivity, k, is obtained by,

$$k = \frac{\mu c_p}{Pr} \tag{2.9}$$

Prandtl Number, Pr, is equal to 0.72 in the current research.

2.2.1 Ideal Gas Relations

For an ideal gas, it is assumed that internal energy is a function of temperature, T, only,

$$e_I = c_v T \tag{2.10}$$

where T is temperature. Substituting the caloric equation of state (2.10) and equation of state of ideal gas,

$$p = \rho R T \tag{2.11}$$

into equation (2.3) results in a new relation for the total energy base on pressure and kinematic energy,

$$\rho e = \frac{p}{(\gamma - 1)} + \frac{1}{2}\rho u_i^2 \tag{2.12}$$

introducing R and γ as the specific gas constant and the ratio of specific heats,

$$R = c_p - c_v \tag{2.13}$$

$$\gamma = \frac{c_p}{c_v} \tag{2.14}$$

where c_v and c_p are specific energy in constant volume and constant pressure, respectively.

In this work, we assume that the constant values $c_v = 7.17\ [(kN.m)/(kg.K)]$ and $c_p = 1.004\ [(kN.m)/(kg.K)]$

2.2.2 Non-Dimensional Form

We use a non-Dimensional form of the Navier-Stokes equations is solved. We non-dimensionalize the flow parameters with the reference values listed in Table 2.2.

Table 2.2: Non-Dimensional parameters and the reference values

Parameter	Dimensional	Non-Dimensional	Reference Value
Length	l	$l^N = \frac{l}{l_{ref}}$	l_{ref}
Density	ρ	$\rho^N = \frac{\rho}{\rho_{ref}}$	ρ_∞
Velocity	$\boldsymbol{u}$	$\boldsymbol{u}^N = \frac{\boldsymbol{u}}{\boldsymbol{u}_{ref}}$	u_∞
Pressure	p	$p^N = \frac{p}{p_{ref}}$	$\rho_\infty c_\infty^2/\gamma$
Time	t	$t^N = \frac{t}{t_{ref}}$	l_{ref}/u_∞
Temperature	T	$T^N = \frac{T}{T_{ref}}$	T_∞
Viscosity	μ	$\mu^N = \frac{\mu}{\mu_{ref}}$	$(\rho_\infty u_\infty l_{ref})/Re$

In table 2.2 Re is Reynolds number and it is defined as $\frac{\rho u l}{\mu}$, and c_∞ is speed of sound at free stream. Non-dimensional parameters for the rest of variables such as thermal conductivity, total energy, and etc. can be extracted by using defined the variable in table 2.2. Changing the dimensional parameters to Non-Dimensional variables inside equation (2.2) results in a Non-dimensional conservative form of the governing equations,

$$
\boldsymbol{W}^N = \begin{bmatrix} \rho^N \\ \rho^N u_1^N \\ \rho^N u_2^N \\ \rho^N u_3^N \\ \rho^N e^N \end{bmatrix}, \quad {}^c\boldsymbol{F}_i^N = \begin{bmatrix} \rho^N u_i^N \\ \rho^N u_i^N u_1^N + \frac{p^N}{\gamma M^2}\delta_{i1} \\ \rho^N u_i^N u_2^N + \frac{p^N}{\gamma M^2}\delta_{i2} \\ \rho^N u_i^N u_3^N + \frac{p^N}{\gamma M^2}\delta_{i3} \\ \rho^N u_i^N h^N \end{bmatrix},
$$

$$
{}^v\boldsymbol{F}_i^N = \begin{bmatrix} 0 \\ -\frac{1}{Re}\sigma_{i1}^N \\ -\frac{1}{Re}\sigma_{i2}^N \\ -\frac{1}{Re}\sigma_{i3}^N \\ -\frac{1}{Re}(u_1^N\sigma_{1i}^N - u_2^N\sigma_{2i}^N - u_3^N\sigma_{3i}^N) + \frac{1}{Re}\frac{1}{Pr}\frac{\gamma}{\gamma-1}Q_i^N \end{bmatrix} \tag{2.15}
$$

Hereafter, for simplicity we eliminate the superscript "N".

2.3 Large Eddy Simulation Model

Large Eddy Simulation (LES) is applied by imposing a high-pass scale filter to eliminate eddies smaller than a threshold. Scale's separation is performed with a convolution kernel filter. In the general case, this filter is applied both in space and time,

$$
\bar{\phi}(\boldsymbol{x}, t) = \frac{1}{\Delta}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(\frac{\boldsymbol{x}-\boldsymbol{\xi}}{\Delta}, t-t')\phi(\boldsymbol{\xi}, t')dt'd^3\boldsymbol{\xi} \tag{2.16}
$$

where ϕ can be any flow parameters and $\bar{\phi}$ is the filtered value of the same parameter. G is the convolution filter and there are three classical filters: box filter, Gaussian filter, and sharp cut off filter [36]. In the present work the first one is utilized,

$$
G(\boldsymbol{x}-\boldsymbol{\xi}) = \begin{cases} \frac{1}{\Delta} & \text{if } |\boldsymbol{x}-\boldsymbol{\xi}| \leq \frac{\Delta}{2} \\ 0 & \text{otherwise} \end{cases} \tag{2.17}
$$

when the box filter is used as the convolution filter, the filtered parameters is given by,

$$\bar{\phi}(\boldsymbol{x},t)=\frac{1}{\Delta}\int_{-\infty}^{\infty}G(\frac{\boldsymbol{x}-\boldsymbol{\xi}}{\Delta})\phi(\boldsymbol{\xi},t)d^3\boldsymbol{\xi} \tag{2.18}$$

Considering one-dimensional case and the change of variable $z=(x-\xi)/\Delta$ the integral (2.18)

$$\bar{\phi}(x,t)=\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}}G(z)\phi(x-z\Delta,t)dz \tag{2.19}$$

Flow parameters can be written in the following form,

$$\phi=\bar{\phi}+\phi' \tag{2.20}$$

where ϕ' is called the Sub-Grid part of the flow parameters.

For compressible flows the Favre filter is employed to calculate filtered variables. Density is used as a weight coefficient multiplied to all the flow parameters,

$$\tilde{\phi}=\frac{\overline{\rho\phi}}{\bar{\rho}} \tag{2.21}$$

and the flow parameters are given by,

$$\phi=\tilde{\phi}+\phi'' \tag{2.22}$$

Using the Favre-filtered variables inside the governing equations lead to one the filtered form of the governing equations,

$$\frac{\partial\bar{\rho}}{\partial t}+\frac{\partial\bar{\rho}\tilde{u}_i}{\partial x_i}=0 \tag{2.23}$$

$$\frac{\partial\bar{\rho}\tilde{u}_i}{\partial t}+\frac{\partial\bar{\rho}\tilde{u}_i\tilde{u}_j}{\partial x_j}+\frac{\partial\bar{p}}{\partial x_j}-\frac{\partial\check{\sigma}_{ij}}{\partial x_j}=-\frac{\partial\check{\tau}_{ij}}{\partial x_j}+\frac{\partial}{\partial x_j}(\bar{\sigma}_{ij}-\check{\sigma}_{ij}) \tag{2.24}$$

$$\frac{\partial \bar{\rho}\tilde{E}}{\partial t}+\frac{\partial(\bar{\rho}\tilde{E}+\bar{p})\tilde{u}_j}{\partial x_j}-\frac{\partial \check{\sigma}_{ij}\tilde{u}_i}{\tilde{x}_j}+\frac{\partial \check{q}_j}{\partial x_j}= \\ -\frac{\partial}{\partial x_j}[(\overline{\rho u_j E}-\bar{\rho}\tilde{u}_j\tilde{E})+(\overline{u_j p}-\tilde{u}_j\bar{p})-(\overline{\sigma_{ij}u_j}-\check{\sigma}_{ij}\tilde{u}_j)-(\bar{q}_j-\check{q}_j)] \tag{2.25}$$

Where $\check{\sigma}_{ij}$ and $\check{q}_j$ are given by,

$$\check{\sigma}_{ij}=\mu(\tilde{T})(2\tilde{S}_{ij}-\frac{2}{3}\delta_{ij}\tilde{S}_{kk}) \tag{2.26}$$

and

$$\check{q}_j=-k(\tilde{T})\frac{\partial \tilde{T}}{\partial x_j}. \tag{2.27}$$

By comparing equations (2.23) to (2.25) with the original governing equation (2.2), the right hand sides are added due to the changing variable to the filtered parameters. These terms appears because the convolution operator does not commute with the derivative neither in space or in time. There is no additional term in the continuity equation due to using the Favre filter. Some of these terms can be neglected because of their small influence in compare to the others. But, first of all we need to regroup some of them in the following form [36],

$$(\overline{\rho u_j E}-\bar{\rho}\tilde{u}_j\tilde{E})+(\overline{u_j p}-\tilde{u}_j\bar{p})=c_p Q_j^{SGS}+\mathcal{J}_j \tag{2.28}$$

where Q_j^{SGS} is called Sub-Grid Scale (SGS) heat flux model, according to [65], and it is detailed in the next section, and $\mathcal{J}_j$ is the SGS turbulent diffusion,

$$\mathcal{J}_j=\frac{1}{2}(\bar{\rho}u_j\widetilde{u_i u_i}-\bar{\rho}\tilde{u}_j\tilde{u}_i\tilde{u}_i-\tau_{ii}) \tag{2.29}$$

the third term on the right hand side of equation (2.25) is considered as the SGS viscous diffusion,

$$\mathcal{D}_j=\overline{\sigma_{ij}u_j}-\check{\sigma}_{ij}\tilde{u}_j \tag{2.30}$$

By replacing Eq.2.28 to Eq.2.30 inside the Eq.2.25 the final format of conservative equation for the total energy is obtained,

$$\begin{aligned}\frac{\partial \bar{\rho}\tilde{E}}{\partial t}+\frac{\partial(\bar{\rho}\tilde{E}+\bar{p})\tilde{u}_j}{\partial x_j}-\frac{\partial \check{\sigma}_{ij}\tilde{u}_i}{\partial x_j}+\frac{\partial \check{q}_j}{\partial x_j}=\\ -\frac{\partial}{\partial x_j}[c_p Q_j^{SGS}+\mathcal{J}_j-\mathcal{D}_j-(\bar{q}_j-\check{q}_j)]\end{aligned} \tag{2.31}$$

2.3.1 Sub-grid Scale Models

In order to simulate fluid flows with LES, one needs to solve equations (2.23) to (2.25). However, The right hand side of the momentum and energy equations should be simplified. This task can be done in two steps, considering the significance of each term, and then introduce a model that relates these terms to filtered-parameters. In the following section some models used to tackle the sub-grid scale terms are represented.

Sub-Grid Scale Model for Momentum Equation

Herein the right hand side of the momentum equation is rewritten,

$$\text{Momentum equations right hand side} := -\frac{\partial \check{\tau}_{ij}}{\partial x_j}+\frac{\partial}{\partial x_j}(\bar{\sigma}_{ij}-\check{\sigma}_{ij})$$

the second term is negligible when compared to the first one and the other terms on the left hand side of the momentum equations (2.24). To model the first term, sub-grid scale stress tensor, The SGS stress tensor can be decomposed into three terms,

$$\begin{aligned}\check{\tau}_{ij}&=\rho\overline{\bar{u}_i\bar{u}_j}-\rho\bar{u}_i\bar{u}_j\\ &=\underbrace{(\rho\overline{\bar{u}_i\bar{u}_j}-\rho\bar{u}_i\bar{u}_j)}_{(I)}+\underbrace{\rho\overline{\bar{u}_i u_j'}+\rho\overline{u_i'\bar{u}_j}}_{(II)}+\underbrace{\rho\overline{u_i'u_j'}}_{(III)}\end{aligned} \tag{2.32}$$

Term (I) in equation (2.32) is termed the Leonard stresses. These stresses are only due to the influence of resolved parameters and the fact that $\bar{\bar{\phi}} \neq \bar{\phi}$. In contrary, term (III) represents the stresses that come from as an effect of the sub-filter scale and they are called LES Reynolds stresses. Finally, Cross-stresses is usually used to refer to term (II) and it reflects the stresses that

result from the interaction between resolved and sub-filter scales.

Many methods have been proposed in the last half century, including very simple, but efficient models, such as Smagorinsky [99] or more complicated methods such as dynamic Smagorinsky models [37, 68, 97]. There are also some higher order SGS methods that employs RANS turbulence model to evaluate SGS turbulence viscosity [35]. In the present work the classical Lilly-Smagorinsky is used.

Lilly-Smagorinsky Model

The Leonard stresses, cross-stresses and LES Reynolds stresses can be combined, despite their different natures, and the stress tensor is written,

$$\tau_{ij} = \mu_{SGS}(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}) - \frac{2}{3}\mu_{SGS}\, div\, \tilde{\boldsymbol{u}} \tag{2.33}$$

where μ_{SGS} is the Sub-grid scale turbulence viscosity. The classical Lilly-smagorinsky method is proposed by Joseph Smagorinsky in 1963 [99]. It uses the idea of Prandtl's mixing length $\nu_t = (l_m^2|\frac{\partial U}{\partial y}|)$ to calculate turbulence viscosity. In this model the turbulence viscosity is proportional to the strain rate and a mixing length,

$$\mu_{SGS} = \rho(C_S\Delta)^2|\bar{S}| \tag{2.34}$$

where,

$$|\bar{S}| = \sqrt{2\bar{S}_{ij}\bar{S}_{ij}} \tag{2.35}$$

is the average of the strain rate of resolved parameters. The strain rate is given by,

$$\bar{S}_{ij} = \frac{1}{2}(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}) \tag{2.36}$$

The Smagorinsky constant, C_{SGS}, usually varies between 0.17-0.24. However, this range is obtained based on the decay rates of the turbulent eddies on isotropic flow in an inertial domain of energy spectrum . However, real flows are not isotropic,for example close to the walls large amount

of non-isotropic eddies are introduced in the flow. Usually, it has to be calibrated based on the physics of the problems. There are more complex methods that adapt the Smagorinsky constant based on different parameters in the flow [33]. However, if the value of the Smagorinsky constant is chosen properly along with a sufficiently good mesh resolution, the method can predict the flow parameters very well. In addition, computational cost is affordable. In the current study for wind turbine simulations it is assumed to $C_{SGS} = 0.15$.

The filter size, Δ, represents length scale of the eddies. The cube root of the volume has been used in the present research to determine the filter size,

$$\Delta = (volume\ of\ a\ cell)^{1/3} \tag{2.37}$$

Sub-Grid Scale Model for Energy Equation

The terms due to the influences of Sub-grid scale inside the energy equation are seen on the right hand side of (2.25). Herein , it is written again,

$$\text{Energy equations right hand side} := -\frac{\partial}{\partial x_j}[c_p Q_j^{SGS} + \mathcal{J}_j - \mathcal{D}_j - (\bar{q}_j - \breve{q}_j)] \tag{2.38}$$

where Q_j^{SGS} is the SGS heat flux and it equals to,

$$Q_j^{SGS} = \bar{\rho}(\widetilde{u_j T} - \tilde{u}_j \tilde{T}) \tag{2.39}$$

Sub-grid scale turbulence diffusion , $\mathcal{J}_j$ is the same order of magnitude as SGS heat flux, based on the comparison of DNS and LES results of compressible isotropic turbulence [65]. However, in the current research the turbulence diffusion has been neglected due to low Mach number of the simulated problems and considering that it is one order of magnitude smaller than filtered terms in the right hand side of energy equation. The SGS viscous diffusion , $\mathcal{D}_j$ is one order of magnitude less than the SGS heat flux and the SGS turbulence diffusion terms. Therefore, it can be neglected even for flow with high Mach number. Finally, the difference of $\bar{q}_j$, and Favre filter flux , $\breve{q}_j$, is also very small is not taken into account in the calculation. Consequently, the SGS heat flux is the only

term that has to be modelled [65],

$$Q_j^{SGS} = -C_s \frac{\Delta^2 \bar{\rho} |\tilde{S}|}{Pr_T} \frac{\partial \tilde{T}}{\partial x_j} \tag{2.40}$$

2.3.2 Near-wall Treatment for Large Eddy Simulation

Although the standard Lilly-Smagorinsky is widely used and results in accurate outcome in the free turbulent flow such as wakes, jet flows, and mixing lengths. For bounded problem or when the simulation includes solving the problem in the near-wall region, the method fails to predict satisfactory results. To remedy from this drawback two widely used methods are demonstrated in the current dissertation. The Van-Driest damping function and the wall-adapting local eddy viscosity method (WALE).

Van Driest wall Adapting Method

As we know, according to Newtonian fluid theory, fluid particles stick to the wall. Therefore, the fluctuations of the flow properties and turbulent viscosity (here eddy viscosity) on the wall should be zero. However, equation (2.34) predicts non-zero value on the wall since Δ and C_s are non-zero values, also the averaged stain rate, $|\bar{S}|$, is not zero on the wall. Then, we need to modify (2.34) in the way that eddy viscosity is gradually vanished while approaching the wall. According to the boundary layer theory [96], describing the relation of eddy viscosity to distance to the wall is proportional to $O(y)^3$, where y is the distant from the closest wall. It can be shown that, [79], the average strain rate defined by (2.36) is $O(y)$. Hence, a damping function is required to mimic the behaviour of Reynolds stress in the near-wall region. A widely used technique is the modification of the the Smagorinsky constant or the filter size (Δ) by the Van driest wall function [113] $[1 - exp(-y/A)]$. In this work we adjust the Smagorinsky constant based on the distant from the wall,

$$C_{SM} = C_S(1 - exp((y^+)^3/A^3))^{0.5} \tag{2.41}$$

wherein c_{SM} is the modified Smagorinsky constant, A is a constant that was first introduce by

Stokes [104] for an oscillating plate and it is a function of kinematic viscosity and the frequency of oscillations. In the current work we assume $A = 26$. Finally y^+ is the non-dimensionalized distance from the nearest wall,

$$y^+ = \frac{yu_\tau}{\nu} \tag{2.42}$$

where u_τ is friction velocity on the wall,

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} \tag{2.43}$$

Wall-Adapting Local Eddy-Viscosity Model

Using the Van Driest function is less expensive compared to the dynamic model, which requires applying LES simulation on two different filter sizes. However, using the strain rate in order to calculate the eddy viscosity causes inaccurate capturing of the energy dissipation in the regions with eddies. Nicoud and Ducros [79] suggested a new method to express the eddy viscosity based on the square of the velocity gradient tensor instead of the strain rate. Their method is well known as Wall-Adapting Local Eddy-Viscosity model (WALE). They claimed that their method not only can predict the asymptotic behaviour of the eddy viscosity in the near-wall region but also it may predict the transition from laminar to turbulent flow. The new introduced term is,

$$\nu_{SGS} = (C_w\Delta)^2 \frac{(S^d_{ij}S^d_{ij})^{(3/2)}}{(\tilde{S}_{ij}\tilde{S}_{ij})^{5/2} + (S^d_{ij}S^d_{ij})^{(5/4)}} \tag{2.44}$$

where $C_w \sim \sqrt{10.6}C_S$ and S^d_{ij} is the traceless part of the square of the Favre-filtered velocity gradient, and

$$S^d_{ij} = \tilde{S}_{ik}\tilde{S}_{kj} + \tilde{\Omega}_{ik}\tilde{\Omega}_{ik} - \frac{1}{3}\delta_{ij}[\tilde{S}_{mn}\tilde{S}_{mn} - \tilde{\Omega}_{mn}\tilde{\Omega}_{mn}] \tag{2.45}$$

where $\tilde{\Omega}_{mn}$ is vorticity correspond to filtered velocity. Furthermore,

$$\tilde{\Omega}_{ij} = \frac{1}{2}(\frac{\partial\tilde{u}_i}{\partial x_j} - \frac{\partial\tilde{u}_j}{\partial x_i}) \tag{2.46}$$

Herein the final conservative form of the governing equations is written, as;

$$\widetilde{\boldsymbol{W}} = \begin{bmatrix} \bar{\rho} \\ \bar{\rho}\tilde{u}_1 \\ \bar{\rho}\tilde{u}_2 \\ \bar{\rho}\tilde{u}_3 \\ \bar{\rho}\tilde{e} \end{bmatrix}, \quad {}^{c}\widetilde{\boldsymbol{F}}_i = \begin{bmatrix} \bar{\rho}\tilde{u}_i \\ \bar{\rho}\tilde{u}_i\tilde{u}_1 + \tilde{p}\delta_{i1} \\ \bar{\rho}\tilde{u}_i\tilde{u}_2 + \tilde{p}\delta_{i2} \\ \bar{\rho}\tilde{u}_i\tilde{u}_3 + \tilde{p}\delta_{i3} \\ \bar{\rho}\tilde{u}_i\tilde{h} \end{bmatrix},$$

$${}^{v}\widetilde{\boldsymbol{F}}_i = \begin{bmatrix} 0 \\ -\frac{1}{Re}(\tilde{\sigma}_{i1} + \tau_{i1}) \\ -\frac{1}{Re}(\tilde{\sigma}_{i2} + \tau_{i2}) \\ -\frac{1}{Re}(\tilde{\sigma}_{i3} + \tau_{i3}) \\ -\frac{1}{Re}(\tilde{u}_1\tilde{\sigma}_{1i} - \tilde{u}_2\tilde{\sigma}_{2i} - \tilde{u}_3\tilde{\sigma}_{3i}) + \frac{1}{Re}\frac{1}{Pr}\frac{\gamma}{\gamma-1}\widetilde{Q}_i + \frac{1}{Re}Q_j^{SGS} \end{bmatrix} \tag{2.47}$$

Chapter 3

Numerical Methodology

3.1 Introduction

Numerical methods have been very effective for engineers and scientists in the past half century in the area of fluid dynamics. Using more powerful machines and improving the accuracy of numerical schemes have enabled scientists to simulate accurately many problems in the field of fluid mechanics.

The Differential form of the governing equations of the Newtonian flow was detailed in the previous chapter. In order to solve the conservative form of the Navier-Stokes equations a numerical, scheme has to be employed. Numerical schemes typically contains the following steps: making a computational domain that contains nodes and cell; discretization of the Navier-Stokes equations in space and time; using numerical methods to solve linear system of equations made of discretized equations of all cells or nodes; and anticipating flow fields and quantities of interest.

The procedure of converting a physical domain into a computational domain is called mesh generation. Computational domain includes a finite number of vertices and elements. Mesh can be generated by commercial software such as GAMBIT, ICEM CFD, ANSYS Mesher, etc.

Discretization of the N-S equations in space and time can be handled with a wide variety of schemes. Many of them are found in Computational Fluid Dynamics reference books [4, 42, 110,

114].

In the current project a finite-volume finite-element method has been used to solve the Navier-Stokes equations. Temporal and convective terms are discretized base on a finite-volume scheme, while the viscous term and source term are approximated with a finite-element method. Discretization of equations and calculation of fluxes result in a linear system of equations on the vertices. In order to solve the linear system of equation a Generalized Minimal RESidual (GMRES) Method has been used.

In this chapter first, the discretized form of the N-S equations is presented. Then, the procedure of solving convective, diffusive, and temporal term is described.

3.2 Navier-Stokes Equations Discretization

Compressible Navier-Stokes equations are written,

$$\frac{\partial \boldsymbol{W}}{\partial t} + \nabla.({}^{c}\boldsymbol{F} + {}^{v}\boldsymbol{F}) = \mathbf{S} \tag{3.1}$$

As it was defined in the previous chapter $\boldsymbol{W}$, ${}^{c}\boldsymbol{F}$, and ${}^{v}\boldsymbol{F}$, represent vectors of conservative variables, convective flux, and diffusive flux, respectively. $\mathbf{S}$ represents the source term vector. A weak form of the governing equations is required for the hybrid finite-element finite volume scheme. It is built by multiplying a test function to each term and integration over the entire domain. Then, the weak form of equation (3.1) is written as,

$$\int_{\Omega} \phi \frac{\partial \boldsymbol{W}}{\partial t} d\Omega + \int_{\Omega} \phi \nabla.{}^{c}\boldsymbol{F} d\Omega + \int_{\Omega} \phi \nabla.{}^{v}\boldsymbol{F} d\Omega = \int_{\Omega} \phi \mathbf{S} d\Omega \tag{3.2}$$

where Ω is the computational domain and ϕ represents the test function. Equation (3.2) is also valid on every cell of the domain. Finite element cells are same as elements as mesh. However, the dual mesh for the finite volume cells are build around each node using the middle points of connected edges to the vertex and center of gravity of cells that encompass the node. A sample of tetrahedral finite-element cell is depicted in Figure 3.1. The contribution of an element to the finite-volume cell has been demonstrated in Figure 3.2.

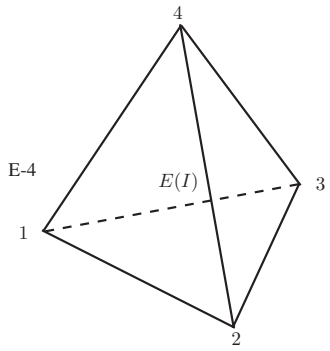


Figure 3.1: Tetrahedral element

The N-S equations can be written on each finite-volume or finite-element cell, as,

$$\int_{C(I)} M_I \frac{\partial \boldsymbol{W}}{\partial t} dV + \int_{C(I)} M_I \nabla .^c \boldsymbol{F} dV + \int_{E(I)} \nabla .^v \boldsymbol{F} N_I dV = \int_{E(I)} \mathbf{S} N_I dV \tag{3.3}$$

where $C(I)$ is a finite-volume cell which encompass the i 'th node, and $E(I)$ is the finite-element cell which has i'th node as one of its vertex. M_I and N_I are weight functions that build the weak form of the Partial Differential Equations (PDEs) on finite-volume and finite element cell, receptively. Since a finite volume scheme is considered for temporal and convective calculation M_I is given by,

$$M_I = \begin{cases} 1 & : \text{on cell C(I)}, \\ 0 & : \text{elsewhere.} \end{cases} \tag{3.4}$$

The finite-element part uses a second order Galerkin method. In the Galerkin method, the basis function is the same as the weight function that makes the weak form of the differential equations.

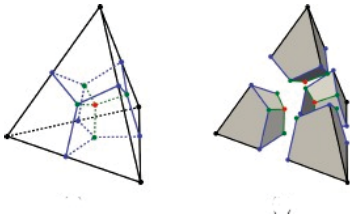


Figure 3.2: Dual mesh on tetrahedral cell [118]

Then, N_I is given by,

$$N_I = \begin{cases} 1 & : \text{at node I,} \\ 0 & : \text{Other nodes.} \end{cases} \tag{3.5}$$

Replacing corresponding weight functions inside equation (3.3) and applying the divergence theorem,

$$\begin{aligned} \int_v (\nabla.\mathbf{F})dV &= \int_{\partial v} \mathbf{F}.dS, \\ \int_v \mathbf{F}.(\nabla g) + g(\nabla.\mathbf{F})dV &= \int_{\partial v} g\mathbf{F}.dS, \end{aligned} \tag{3.6}$$

leads to the convection and the diffusion term results,

$$\begin{aligned} \int_{C(I)} \frac{\partial \mathbf{W}}{\partial t} dV + \int_{\partial C(I)} {}^{c}\mathbf{F}(W_h).\vec{n}_{\partial C(I)} dS = \\ + \int_{E(I)} {}^{v}\mathbf{F}(W_h).\nabla N_I dV - \int_{\partial E(I)} {}^{v}\mathbf{F}(W_h).\vec{n}_{\partial C(I)} N_I dS + \int_{E(I)} \mathbf{S}(W_h) N_I dV \end{aligned} \tag{3.7}$$

Now, the finite-volume method for temporal and convective terms and finite-element scheme for the viscous and source term is applied.

3.2.1 Viscous flux calculation

According to equation (3.7), the viscous flux contains two parts,

$$\text{viscous flux} := -\int_{E(I)} {}^{v}\mathbf{F}(W_h).\nabla N_I dV + \int_{\partial E(I)} {}^{v}\mathbf{F}(W_h).\vec{n}_{\partial C(I)} N_I dS$$

The first term represents the integration of fluxes over a finite-element cell and the second term takes into account the changes on the boundary of the element. Since the viscous flux which pass the boundary of a cell is usually very small in compare to the first term, only the first part needs to be evaluated.

For a Lagrangian element, any arbitrary variable inside an element is calculated from the summation of product of value of that variable at each node and its corresponding basis function,

$$W|_{E(J)} = \sum_{j=1}^{N_s} W_j N_j \tag{3.8}$$

where N_s is the number of nodes for each element, for example it equals to 4 for a tetrahedral element, and $E(I)$ includes all finite-element cells that are connected to node I. Introducing,

$$F(W)|_{E(I)} = \sum_{J \in E(I)} F(W)|_{E(J)}, \tag{3.9}$$

then the volume integral of viscous term can be rewritten,

$$-\int_{E(I)} {}^{v}\mathbf{F}(W_h).\nabla N_I dV = -\sum_{J \in E(I)} \int_{E(J)} {}^{v}\mathbf{F}(W_h).\nabla N_{IJ} dV, \tag{3.10}$$

where N_{IJ} is the basis function of the vertex I that corresponds to support element J.

A second order accuracy for the finite-element method requires a linear basis function that satisfies the conditions in equation (3.5). In the case of the tetrahedral element shown in Figure 3.1 a Lagrangian element basis function at each node is given by,

$$N_j = b_j x + c_j y + d_j z + e_j, \quad j = 1, \ldots, 4. \tag{3.11}$$

Equation (3.11) is written at each node and the linear coefficients b to e are evaluated from solving a 4×4 system, for example if $j = 1$ then,

$$\begin{aligned} b_1 x_1 + c_1 y_1 + d_1 z_1 + e_1 &= 1 \\ b_1 x_2 + c_1 y_2 + d_1 z_2 + e_1 &= 0 \\ b_1 x_3 + c_1 y_3 + d_1 z_3 + e_1 &= 0 \\ b_1 x_4 + c_1 y_4 + d_1 z_4 + e_1 &= 0, \end{aligned} \tag{3.12}$$

Similar system of equation can be written for the rest of vertices,

$$\begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \\ e_1 & e_2 & e_3 & e_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.13}$$

Then,

$$\begin{bmatrix} b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \\ e_1 & e_2 & e_3 & e_4 \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix}^{-1}. \tag{3.14}$$

Therefore, first linear base function coefficient are given by,

$$\begin{aligned} b_1 &= -\frac{1}{6V}\begin{vmatrix} 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \\ 1 & y_4 & z_4 \end{vmatrix}, \quad c_1 = \frac{1}{6V}\begin{vmatrix} x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \\ x_4 & 1 & z_4 \end{vmatrix}, \\ d_1 &= -\frac{1}{6V}\begin{vmatrix} x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \end{vmatrix}, \quad e_1 = \frac{1}{6V}\begin{vmatrix} x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \end{vmatrix}. \end{aligned} \tag{3.15}$$

In the same manner the basis coefficients for the other vertices of a tetrahedral are derived. V is the volume of a tetrahedral and it is given by,

$$V = \frac{1}{6}\begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix}. \tag{3.16}$$

According to (3.10), the derivatives of basis function is required rather than its value. Then,

$$\nabla N_j = \begin{bmatrix} b_j \\ c_j \\ d_j \end{bmatrix}. \tag{3.17}$$

Note that the viscous flux equation contains velocity derivatives that has to be constructed over each element. In the following, it is demonstrated for any arbitrary variable u,

$$\begin{bmatrix} \partial u/\partial x \\ \partial u/\partial y \\ \partial u/\partial z \end{bmatrix} = \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}. \tag{3.18}$$

In the next step we build the viscous term introduced in the previous chapter and calculate the integral in equation (3.10). Finally, all the viscous contribution of finite-element cells are summed up to calculate the flux at their supporting vertices.

3.2.2 Source term

The source term is also solved with the finite-element method,

$$\text{source term} := \int_{E(I)} \mathbf{S}(W_h) N_I dV. \tag{3.19}$$

If the source term does not include a gradient, we may simply take the average of value over the support elements,

$$\int_{E(I)} \mathbf{S}(W_h) N_I dV = \frac{1}{N_s} \sum_{J \in E(I)} \frac{1}{4} (S_j(W_h) vol(E(J)) \tag{3.20}$$

3.2.3 Convection flux calculation

In order to calculate the convection part, Roe's method is used. However, since Roe's method results into a first order of accuracy in space, a Monotone Upwinding-centred for Conservative Laws (MUSCL) is employed. Total Variation Diminishing (TVD) is also necessary, since the higher

order of accuracy under certain condition, such as shock or physical discontinuity, may produce some extra non-physical wiggles into the solution. Then Van-Albada scheme is used to control the primitive variables variation slop from nodes to the surfaces of the finite-volume element. Herein, the convective term (Eq.3.7) is written,

$$\text{convection flux} := \int_{C(I)} {}^{c}\boldsymbol{F}.\vec{n}_{\partial C(I)} ds$$

where $\vec{n}_{\partial C(I)}$ is the boundary of finite-volume cell I. Then, the convective flux can be decomposed into,

$$\int_{C(I)} {}^{c}\boldsymbol{F}.\vec{n}_{\partial C(I)} dS = \sum_{J \in k(I)} \int_{C(IJ)} {}^{c}\mathbf{F}(W_h).\vec{n}_{\partial C(IJ)} ds \tag{3.21}$$

$\vec{n}_{\partial C(I)}$ is normal to the common face between cell $C(I)$ and cell $C(J)$. the normal vector and flux are considered constant over the common face, therefore,

$$\int_{C(I)} {}^{c}\boldsymbol{F}.\vec{n}_{\partial C(I)} dS = \sum_{J \in k(I)} {}^{c}\mathbf{F}(W_h)\Big|_{\partial C(IJ)} .\vec{n}_{\partial C(IJ)} S_{\partial C(IJ)}. \tag{3.22}$$

If $\mathbf{F}_{IJ}$ is defined as the product of convective flux and surface's norm,

$$\mathbf{F}_{IJ} = {}^{c}\,\mathbf{F}(W_h)\big|_{\partial C(IJ)}.\vec{n}_{\partial C(IJ)}$$

Then, equation (3.22) is rewritten,

$$\text{convection flux} := \sum_{J \in k(I)} \mathbf{F}_{IJ} S_{\partial C(IJ)} \tag{3.23}$$

Roe's scheme

Roe's method is an approximate solution to Riemann problem that evaluate the fluxes on the common interface between two neighbour cells [93]. A piece-wise constant assumption is considered for primitive variables over a control-volume cell in the first order Roe method. As it is seen in Figure 3.3 , there is a discontinuity in the solution at the interface between two cells.

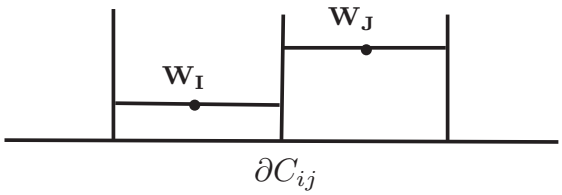


Figure 3.3: First order Roe's method

$$\mathbf{F}_{ij} = \frac{1}{2}[^c\boldsymbol{F}_i + ^c\boldsymbol{F}_j] - \alpha\frac{1}{2}\partial^c\boldsymbol{F}_{ij} \tag{3.24}$$

where,

$$\begin{aligned} \partial^c\mathbf{F}_{ij} &= |A_{ij}|\partial\boldsymbol{W}_{ij} \\ \partial\boldsymbol{W}_{ij} &= \boldsymbol{W}_j - \boldsymbol{W}_i \end{aligned}. \tag{3.25}$$

For Roe's approximation of Riemann solver $\alpha = 1$, and it is called upwind factor and describes later in this chapter. A_{ij} is the Jacobian matrix of convective flux based on the conservative variables. The notation $\bullet_{ij}$ means the average values and according to the Roe method they are obtained by [42],

$$\begin{aligned} \rho_{ij} = \sqrt{\rho_j\rho_j} &= \frac{\rho_i\sqrt{\rho_j} + \rho_j\sqrt{\rho_i}}{\sqrt{\rho_i} + \sqrt{\rho_j}} \\ u_{ij} &= \frac{u_i\sqrt{u_i} + u_j\sqrt{u_j}}{\sqrt{\rho_i} + \sqrt{\rho_j}} \\ h_{ij} &= \frac{h_i\sqrt{h_i} + h_j\sqrt{h_j}}{\sqrt{\rho_i} + \sqrt{\rho_j}} \end{aligned}. \tag{3.26}$$

The average pressure is evaluated using,

$$p_{ij} = (\gamma - 1)\rho_{ij}(h_{ij} - \frac{1}{2}V_{ij}^2) \tag{3.27}$$

and speed of sound at the interface between node i and j is given by,

$$c_{ij}^2 = (\gamma - 1)(h_{ij} - \frac{1}{2}V_{ij}^2) \tag{3.28}$$

Where V_{ij} is the velocity value at the interface.

Roe-MUSCL scheme

A piece-wise linear assumption results in a first order accurate solutions. Although it is stable, this assumption causes inaccurate results wherever discontinuity exists, for example shocks or geometry discontinuities. The MUSCL method is used to increase the accuracy to second order by assuming piecewise linear variation of primitive variable inside each cell. Values of these variables on each side of common surface between two cells can be calculated by extrapolation of the values on the nodes Figure 3.4. Then, the second order Roe's method can be written as,

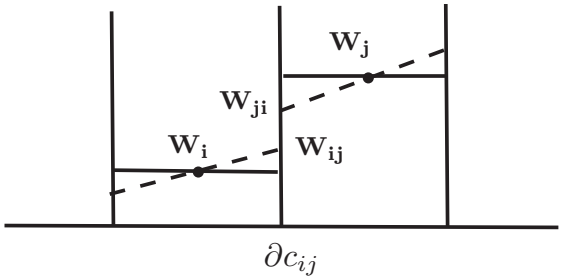


Figure 3.4: Second order Roe's method

$$\frac{1}{2}[{}^{c}\boldsymbol{F}_{ij} + {}^{c}\boldsymbol{F}_{ji}] - \alpha\frac{1}{2}|A_{ij}|\partial\boldsymbol{W}_{ij}. \tag{3.29}$$

In order to calculate ${}^{c}\boldsymbol{F}_{ij}$ and ${}^{c}\boldsymbol{F}_{ji}$, primitive variable $[\rho, u, v, w, p]^T$ have to be calculated on both sides of the interfaces of finite-volume cells, then,

$$\begin{aligned} \boldsymbol{V}_i^+ &= \boldsymbol{V}_i + \frac{1}{4}[(1-k)\boldsymbol{\Delta}_i^u + (1+k)\boldsymbol{\Delta}_{ij}^c], \\ \boldsymbol{V}_j^- &= \boldsymbol{V}_j - \frac{1}{4}[(1-k)\boldsymbol{\Delta}_j^u + (1+k)\boldsymbol{\Delta}_{ij}^c]. \end{aligned} \tag{3.30}$$

The second term on the right hand side controls the variation variable of slope inside cell i and j. The notation $\boldsymbol{\Delta}_i^u$ and $\boldsymbol{\Delta}_{ij}^c$ indicates the amount of upwind and central approximation slope in the cell, respectively. Then, k is a factor that control the distribution of each scheme to the slope. The constant k is set to zero in our simulation, and instead a β-scheme introduce by Camarri et al. [18] is used to control the distribution on the cell and its neighbours to evaluate the flow parameters at

the interface of the cell.

$$\boldsymbol{\Delta}_i^u = 2\nabla\boldsymbol{\Delta}_i^\beta - \boldsymbol{\Delta}_{ij}^c, \tag{3.31}$$

$$\boldsymbol{\Delta}_{ij}^c = \boldsymbol{V}_j - \boldsymbol{V}_j, \tag{3.32}$$

where the operator $\nabla\boldsymbol{\Delta}_i^\beta$ is defined by the β-scheme,

$$\nabla\boldsymbol{\Delta}_i^\beta = \beta\nabla\boldsymbol{\Delta}_i.\vec{s}_{ij} + (1-\beta)\boldsymbol{\Delta}_{ij}^c. \tag{3.33}$$

Amount of $\vec{s}_{ij}$ equals to $\boldsymbol{X}_j - \boldsymbol{X}_i$; $\nabla\boldsymbol{\Delta}_i$ is the average of gradient of the primitive variable $\boldsymbol{V}$ on cell i . A finite-element scheme is used to calculate the average gradient,

$$\begin{aligned} \nabla\boldsymbol{\Delta}_i &= \frac{\int_{C(i)} \nabla(\boldsymbol{V}_i N_i dV)}{\int_{C(i)} dV} \\ &= \frac{1}{vol(C(i))} \sum_{J\in E(I)} \frac{vol(E(J))}{N_s} \sum_{k=1}^{N_s} \boldsymbol{V}_k (\Delta N)_k, \end{aligned} \tag{3.34}$$

where N_s is the number of nodes on each element. For example, for a tetrahedral with a linear basis function it equals to 4.

Van-Albada limiter

Equation 3.29 is not a TVD scheme and can introduce some spurious oscillations into the solution. The Van-Albada limiter is used to limit the slop and primitive flux. It is written as,

$$\begin{aligned} \boldsymbol{V}_i^+ &= \boldsymbol{V}_i + \frac{\phi_{va}(\boldsymbol{\Delta}_i^u, \boldsymbol{\Delta}_{ij}^c)}{4}[(1-k)\boldsymbol{\Delta}_i^u + (1+k)\boldsymbol{\Delta}_{ij}^c], \\ \boldsymbol{V}_j^- &= \boldsymbol{V}_j - \frac{\phi_{va}(\boldsymbol{\Delta}_j^u, \boldsymbol{\Delta}_{ij}^c)}{4}[(1-k)\boldsymbol{\Delta}_j^u + (1+k)\boldsymbol{\Delta}_{ij}^c], \end{aligned} \tag{3.35}$$

where ϕ_{va} is the Van-Albada slope limiter,

$$\phi_{va}(a,b) = \begin{cases} 0 & : \quad \text{if} \quad a.b < 0 \\ \frac{2ab+\epsilon}{a^2+b^2+\epsilon} & : \quad \text{otherwise} \end{cases} \tag{3.36}$$

where ϵ is very small value to ensure that the denominator is not zero. Then, finally equation (3.35) is used to interpolate the flow parameters on the two sides of the interfaces of the cells. These variable are used to calculate the fluxes in the first two terms on the right hand side of equation (3.29).

Artificial dissipation factor

To evaluate the convective flux, it is also necessary to calculate the third term that changes non-linearly with a grid configuration [54],

$$\text{Non-linear Term} := \frac{1}{2}\alpha|\boldsymbol{A}_{ij}|\partial\boldsymbol{W}_{ij} \tag{3.37}$$

$$|\boldsymbol{A}_{ij}| = \boldsymbol{P}_{ij}^{-1}|\boldsymbol{\Lambda}_{ij}|\boldsymbol{P}_{ij} \tag{3.38}$$

where $\mathbf{P}_{ij}^{-1}$, $|\mathbf{\Lambda}_{ij}|$, and $\mathbf{P}_{ij}$ are defined in Appendix A.

The coefficient α is called the upwind factor, artificial dissipation factor, or blending function. Choosing $\alpha = 0$ makes the numerical method a central scheme that is unconditionally unstable and it is not desirable, though it provides a second order accurate scheme. On the other hand, $\alpha = 1$ results in completely Roe-MUSCL method which is a stable method. The later is widely utilized in Reynolds Averaged Navier-Stokes equations(RANS) or in Unsteady-RANS (URANS). However, selecting Roe-MUSCL method along with Large Eddy Simulation models produces extra and non-physical dissipation into the solution. For the sake of accuracy, it is very important to use an upwind factor. The objective is finding the smallest value of α that produces the most accurate solution without stability compensation.

In the current dissertation a wiggle detector method is used to adjust the upwind factor [106] . A wiggle is detected presents if the gradient of the flow parameters along an edge changes twice [23],

$$
\begin{aligned}
(\Phi_i - \Phi_{i-1})(\Phi_{i+1} - \Phi_i) < 0 \\
(\Phi_{i+2} - \Phi_{i+1})(\Phi_{i+1} - \Phi_i) < 0
\end{aligned}
\tag{3.39}
$$

Both conditions in equation (3.39) have to pass in order to declare that a wiggle is found at the vertex. For example according to the wiggle definition, there is a wiggle at node i in Figure 3.5

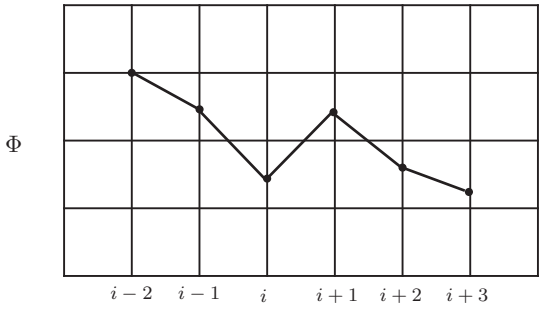


Figure 3.5: Wiggle detection along an edge on an one-dimensional grid.

In two and three dimensional grid, process of finding the gradient is more complex. For the sake of simplicity a 2D grid is demonstrated in Figure 3.6, a similar procedure is applicable in three dimensional domain. Thick lines, in Figure 3.6, represent the boundary of finite-element cells and thin lines define the boundary of dual cell that surround vertices i and j. The following conditions should be satisfied to assert that a wiggle is detected at the common interface between two vertices,

$$
\begin{aligned}
[(\boldsymbol{\Delta}\Phi)^L_{ij}.\vec{n}_{ij}]\frac{(\phi_j - \phi_i)}{(\mathbf{X}_j - \mathbf{X}_i)} < 0 \\
[(\boldsymbol{\Delta}\Phi)^R_{ij}.\vec{n}_{ij}]\frac{(\phi_j - \phi_i)}{(\mathbf{X}_j - \mathbf{X}_i)} < 0
\end{aligned}
\tag{3.40}
$$

where $(\boldsymbol{\Delta}\phi)^c_{ij}.\vec{n}_{ij} = (\phi_j - \phi_i)/(\mathbf{X}_j - \mathbf{X}_i)$ is central gradient along edge ij. The central gradient corresponds to $\Phi_{i+1} - \Phi_i$ term in the non-equalities (3.39). Therefore, $(\boldsymbol{\Delta}\phi)^L_{ij}.\vec{n}_{ij}$ and $(\boldsymbol{\Delta}\phi)^R_{ij}.\vec{n}_{ij}$ are upwind and downwind gradients of ϕ along the edge that connect node i to node j, respectively. We need to find the support elements that the extension of edge passes through them. These are marked as R and L in Figure 3.6. Gradient $(\boldsymbol{\Delta}\phi)^L_{ij}$ and $(\boldsymbol{\Delta}\phi)^R_{ij}$ are calculated from the finite-element method defined in equation (3.18).

Since eddies in the wall region get smaller the rapid change of the gradients can be physical

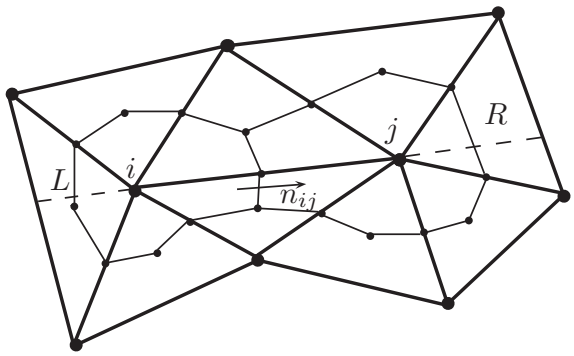


Figure 3.6: Wiggle detection along an edge on a 2-dimensional grid.

explained in that area. Then, detecting the wiggles base on equation (3.40) may cause to over-dissipation in the near-wall region. To avoid the inevitable situation or decrease the over-dissipation, parameter θ is introduced as a threshold, Then, non-equalities (3.40) is modified to,

$$\begin{aligned} [(\boldsymbol{\Delta}\Phi)_{ij}^{L}.\vec{n}_{ij}]\frac{(\phi_j-\phi_i)}{(\mathbf{X}_j-\mathbf{X}_i)} &< \theta < 0 \\ [(\boldsymbol{\Delta}\Phi)_{ij}^{R}.\vec{n}_{ij}]\frac{(\phi_j-\phi_i)}{(\mathbf{X}_j-\mathbf{X}_i)} &< \theta < 0 \end{aligned} \tag{3.41}$$

In [106] is shown that a very small negative value for θ ($\approx$ between -0.00001 to -0.0001) is a good choice to vanish the non-physical wiggles inside the flow field without compensating the eddies inside the near-wall region.

If condition (3.41) is satisfied, upwinding factor should be increased a little to vanish the wiggles or at least decrease the quantities in the left hand side of non-equalities (3.41). In contrast, if no wiggle is detected α is increased. The increment and decrement are implemented by a linear function,

$$\begin{aligned} \theta - \min[\boldsymbol{\Delta}_i^u.\frac{\boldsymbol{\Delta}_{ij}^c}{(\boldsymbol{X}_j-\boldsymbol{X}_i)}] \\ \max[\boldsymbol{\Delta}_i^u.\frac{\boldsymbol{\Delta}_{ij}^c}{(\boldsymbol{X}_j-\boldsymbol{X}_i)}] - \theta \end{aligned} \tag{3.42}$$

3.3 Boundary Conditions

In the previous sections, the discretization technique to approximate the Navier-Stokes equations have been described. When a finite element-finite volume technique is used in order to solve numerically PDE's such as the Navier-stokes equations numerically, boundary condition should be

impose correctly. Defining a boundary condition should represent the physical boundary condition. Some boundary conditions are dictated by the physical conditions. It is also important that the discretization method that imposed on the boundaries be consistent with the order of accuracy of the internal scheme on the discretized equations, and does not deteriorate the stability.

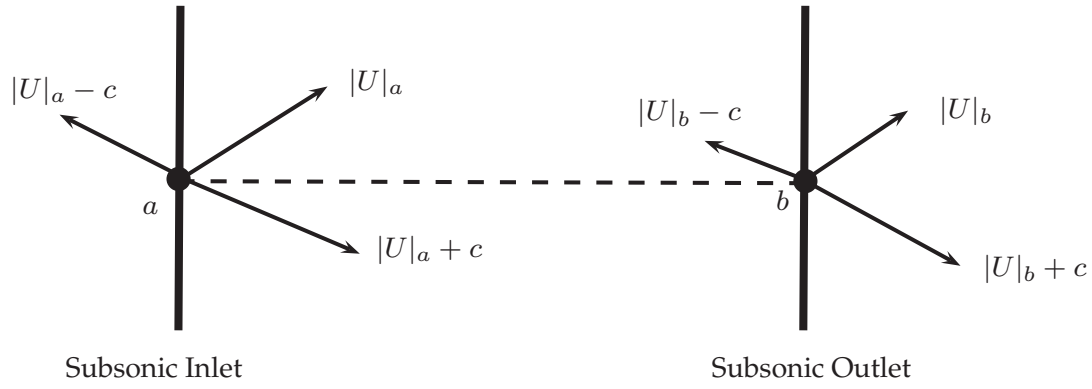
In this chapter, first we will discuss the number of physical boundary conditions and those who needs to calculated from the interior domain. Then, we present the boundary conditions in our in-house code and their implementation.

3.4 Boundary conditions in supersonic and Subsonic Flows

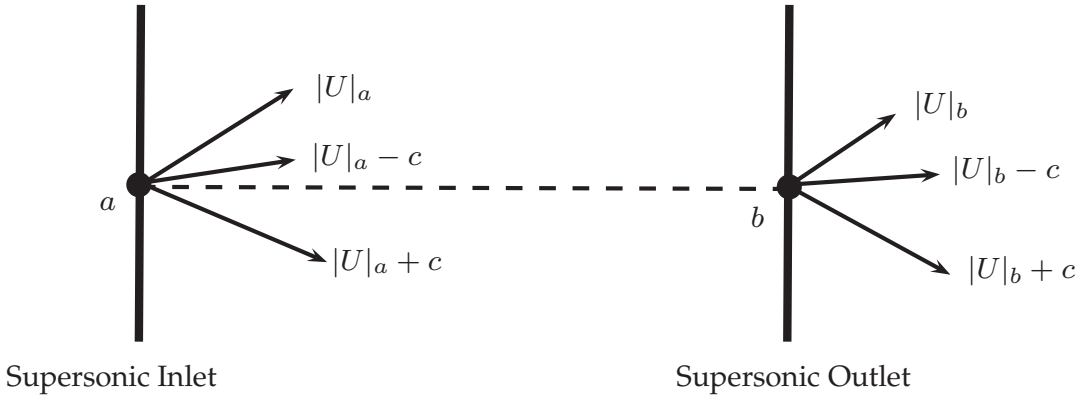
Information inside a stationary medium propagate with the speed of sound. In numerical methods, it is very important to identify the domain of dependency and range of influences at each point. Inside a inviscid flow, direction and speed in which information travel inside the medium are given according to the eigenvalues of the conservative flux. The direction of the eigenvalues determine which primitive variables (or conservative variables) have to be imposed on a boundary and which parameters has to be interpolated from the interior node.

For an Euler flow, number of physical boundary conditions that should be imposed on a boundary are defined by the eigenvalue of the conservative flux at individual points. For a three-dimensional inviscid flow, eigenvalues are defined in equation (3.53). Therefore, three of them have a same value as absolute velocity at each node $|U|$, other two are given by $|U|+c$ and $|U|-c$. For one-dimensional flows, this is shown in 3.7, and the conservative variables are $\{\rho \quad \rho u \quad \rho E\}$. For example consider the inlet boundary in 3.7a.

It shows that for a subsonic flow on a inlet boundary two eigenvalues are in the positive direction, and one in the negative direction. Direction of eigenvalues represents the influence zone at the boundary node and usually its neighbours. It means that the region of dependency of two out of three variables is only limited to the outside the computational domain, while the domain of dependency for the third parameter is inside the computational domain. In the other word, two out of three conservative variables should be imposed by the physical boundary condition, whereas



(a) Subsonic Flow



(b) Supersonic Flow

Figure 3.7: Eigenvalue directions for the inlet and outlet boundary condition on subsonic and supersonic flows.

the last is obtained from the numerical calculation from the interior domain. On the other hand, at the outlet boundary condition in 3.7a, the region of dependency corresponds to two conservative variables related to the inside of computational domain,and obtained from the numerical estimation. The third variables should be imposed based on the physical conditions.

Similar explanations can be used for the boundary condition illustrated in 3.7b. Therefore, in a supersonic flow, all the conservative are implemented according to the physical boundary condition at the inlet. In contrary, the conservative variables should be calculated from the numerical schemes. The methodology, also, can be extended to 3-dimensional flows. The summary of the number of physical or numerical variables ones are reported in table 3.1

Table 3.1: Number of physical and numerical conservative variables for 3-dimensional flows

	subsonic		supersonic	
	Inlet	outlet	Inlet	Outlet
physical variables	4	1	5	0
numerical variables	1	4	0	5

3.5 No-Slip Boundary Condition

No-slip boundary condition is used on solid bodies. It means that the relative velocity of the fluid on the surface is zero. Therefore, flow neither enters nor exits from the no-slip surface, nor any tangential velocity is presented. This is a Drichlet boundary condition.

3.6 Periodic Boundary Condition

In 2.5-D simulation, periodic boundary condition is essential in the span-wise direction. Implementation of the periodic boundary condition is different from the rest of boundaries. Using a periodic boundary condition is very similar to communication at the interface of partitions in parallel code. A 2-dimensional mesh is shown in Figure 3.8. Node i and j are located on a periodic boundary, and node k and l are located on the pair of that surface. Total fluxes that transfer between node i and j should be equal to that passes at the interface of node k and l. The amount of flux is summation of both. This is very similar to the way that information between is exchanged between partitions for a parallel program.

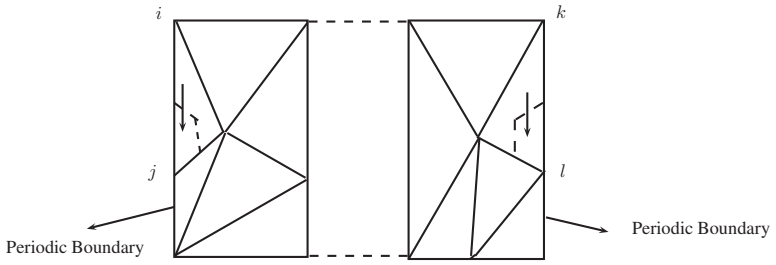


Figure 3.8: Periodic boundary condition on a pair of faces.

3.6.1 Time Discretization

Large eddy simulation models requires at least second order accurate in time discretization. Then an implicit second order method is implemented to discretize the temporal term. First, we rewrite equation (3.7) by keeping time integral in the left and taking other term to the right hand side of equation,

$$\int_{C(I)} \frac{\partial \boldsymbol{W}_i}{\partial t} dV = K(\boldsymbol{W}_i) \tag{3.43}$$

Function K contains both convection and diffusion fluxes. By assuming a constant $\boldsymbol{W}_i$ over each cell at time n the left hand side of equation (3.43) is changed to,

$$\int_{C(I)} \frac{\partial \boldsymbol{W}_i}{\partial t} dV = \frac{\partial \boldsymbol{W}_i}{\partial t}\Big|^n vol(C(i)) \tag{3.44}$$

In an implicit method the right hand side in equation (3.43) also has to be evaluated at one advance time, Then,

$$\frac{\partial \boldsymbol{W}_i}{\partial t}\Big|^n vol(C(i)) = \boldsymbol{K}(\boldsymbol{W}_i^{n+1}) \tag{3.45}$$

Using the Taylor expansion of the right hand side about time n results in,

$$\frac{\partial \boldsymbol{W}_i}{\partial t}\Big|^n vol(C(i)) = K(\boldsymbol{W}_i^n) + \frac{\partial \boldsymbol{K}_i^n}{\partial t}\Delta t + O(\Delta t)^2 \tag{3.46}$$

By chaining rules,

$$\frac{\partial \boldsymbol{K}_i^n}{\partial t} = \frac{\partial \boldsymbol{K}_i}{\partial \boldsymbol{W}_i}\Big|^n \frac{\partial \boldsymbol{W}_i^n}{\partial t}\Big|^n. \tag{3.47}$$

By substituting the right hand of equation (3.47) inside equation (3.46) and rearranging the terms we get,

$$\Big[vol(C(i)) - \frac{\partial \boldsymbol{K}_i}{\partial \boldsymbol{W}_i}\Big|^n \Delta t\Big] \frac{\partial \boldsymbol{W}_i}{\partial t}\Big|^n = \boldsymbol{K}_i^n \tag{3.48}$$

If a first order accurate method in time is used,

$$\frac{\partial \boldsymbol{W}_i}{\partial t}\Big|^n = \frac{\boldsymbol{W}_i^{n+1} - \boldsymbol{W}_i^n}{\Delta t}, \tag{3.49}$$

while if the second order scheme is used to discretize the temporal time, the following form is used,

$$\begin{aligned} \frac{\partial \boldsymbol{W}_i}{\partial t}\Big|^n &= \frac{((1+2\tau)/(1+\tau))\boldsymbol{W}_i^{n+1} - (1+\tau)\boldsymbol{W}_i^n + (\tau^2/1+\tau)\boldsymbol{W}_i^{n-1}}{\Delta t} \\ &\equiv \frac{((1+2\tau)/(1+\tau))\delta\boldsymbol{W}_i^n - (\tau^2/1+\tau)\delta\boldsymbol{W}_i^{n-1}}{\Delta t} \end{aligned} \tag{3.50}$$

where $\tau = \Delta t_n / \Delta t_{n-1}$ and

$$\delta \boldsymbol{W}_i^n = \boldsymbol{W}_i^{n+1} - \boldsymbol{W}_i^n \tag{3.51}$$

3.7 Time Advancement

Time restriction is very important factor to ensure the stability when an explicit method is used. There is a strict Courant-Friedrich-Lewy (CFL) condition that defines the maximum time step [22]. The CFL number for the one-dimensional wave equation is given by,

$$CFL = \frac{a\Delta t}{\Delta x}. \tag{3.52}$$

Where a is the wave speed; Δx and Δt are step time and cell size, respectively. Maximum time step should be equal or less than the required time to transport information from one cell to its neighbors in order to guarantee the stability of a scheme. Maximum CFL number varies from one approach to another based on the order of accuracy and type of time discretization scheme.

In contrary, an implicit approach is unconditionally stable for the wave equation. However, we have to be careful to extend it to Navier-Stokes equations or any non-linear PDE equations. The stability analysis are usually derived on linear PDEs. Then, some concerns regrading the stability rise whereas a solution to non-linear PDEs such as Euler or Navier-Stokes equations is sought.

That imposes some limitations on the time step size for an implicit approach. Time step constraint depends on physical condition of the problem, grid size and quality, discretization methods and the linear solver approach.

In general there are are two types of CFL number to consider, acoustic and advective CFL number . Acoustic CFL number (cdt/dx) is calculated based on speed of sound velocity inside the medium. On the other hand, advective CFL number $(|V|dt/dx)$ uses the local velocity to calculate the Courant number.

Herein we discussed how time step is calculated in our code. For the Euler equations, the stability analysis can be defined based on convection. The information is propagate according to the eigenvalues of the convection Jacobian. For a there dimensional Euler equation, Jacobian are given by,

$$\Lambda_i = \begin{bmatrix} |V|_i & 0 & 0 & 0 & 0 \\ 0 & |V|_i & 0 & 0 & 0 \\ 0 & 0 & |V|_i & 0 & 0 \\ 0 & 0 & 0 & |V|_i - c_i & 0 \\ 0 & 0 & 0 & 0 & |V|_i - c_i \end{bmatrix}, \tag{3.53}$$

where $|V|_i$ is the absolute velocity at individual nodes and c_i is their corresponding speed of sound. Therefore, for an inviscid flow time step based on the spectral radius or the maximum eigenvalues, $\lambda_{max} = |V|_i + c_i$, is given by,

$$\Delta t_{conv} = \frac{CFL}{(\Delta S_x + \Delta S_y + \Delta S_z)\lambda_{max}}, \tag{3.54}$$

where ΔS_x, ΔS_y, and ΔS_z are components of the face vector.

In the case of diffusion equation considering the pressure viscous time step is considered,

$$\Delta t_{diff} = \frac{CFL}{C.\max(\frac{\gamma}{Pr}, 1.0)\frac{Re}{\rho}(\Delta S_x^2 + \Delta S_y^2 + \Delta S_z^2)}, \tag{3.55}$$

where C is a constant and it is usually take a value between $1 \leq C \leq 4$ [13]. We use $C = 2$ in

our code. In the Navier-stokes equation both convective and diffusion terms have to be considered. Therefore following equation to calculate the time step is used,

$$\Delta t = \frac{CFL}{(\Delta S_x + \Delta S_y + \Delta S_z)\lambda_{max} + C.\max(\frac{\gamma}{Pr}, 1.0)\frac{Re}{\rho}(\Delta S_x^2 + \Delta S_y^2 + \Delta S_z^2)} \tag{3.56}$$

This relation is calculated on all nodes on the domain, and the minimum value over the entire domain is considered as the global time step to advance the computation in time. Another approach is to calculate the minimum of acoustic (convective) time step and viscous time step separately, and then take the minimum value between those value, $\Delta t = \min(\min_{\in\Omega} \Delta t_{conv}, \min_{\in\Omega} \Delta t_{diff})$.

3.8 Preconditioning Low Mach Flow

We use a compressible form of the conservative equations to solve the flow with our in-house code. However, if a compressible equation is employed to solve a low Mach number flow ($M < 10^{-2}$), the system of equations is very stiff. That is due to a large discrepancy between the acoustic time scale and the convective time scale [9]. Acoustic time scale is the time that information travels inside a medium, while, the convective time scale represents the speed of fluid particles. It also can be seen in pressure term of the non-dimensional form of equation (2.15). This term goes to infinity as Mach number goes to zero, and that makes the system stiff.

Turkel [111] introduced a low Mach preconditioner for compressible flows. The method is called Roe-Turkel, and uses a diagonal Matrix ($(Diag(\beta^2, 1, 1, 1, 1))$ to improve the convergence of the system. More details about implementing the preconditionar may found in reference [115].

3.9 Linear Solver

Finally, we have a sparse linear system of equations that needs to be solved. A Linear system which is built upon a one-to-one mapping. Each row of the matrix represents the discretized equation of a grid point. Consequently, all the matrix coefficients, on a row, are zero except those corresponding to the node itself and those that share an edge with it. The maximum number of

neighboring nodes barely exceeds 50 ,for a medium to bad quality mesh, on an unstructured 3-dimensional grid, while total number of nodes could be in order of millions. Thus, using a sparse linear solver is a necessary condition due to memory restrictions. Still, solving the linear system is possibly the most computationally demanding task for CFD codes. Therefore, there are many researchers in the field of computer science that work on improving the efficiency of solvers. Some of the most robust sparse linear solver package can be found in PETSc [5], HYPER [30], and MUMPS [3]. Herein, we try to shed some light on the methodology of linear solver used in our in-house code.

Mixing equations 3.46 and 3.49 results in to the final linear system that need to be solved:

$$\left[\frac{vol(C(i))}{\Delta t} - \frac{\partial \boldsymbol{K}_i}{\partial \boldsymbol{W}_i}\Big|^n\right](\boldsymbol{W}_i^{n+1} - \boldsymbol{W}_i^n) = \boldsymbol{K}_i^n \tag{3.57}$$

The system (3.57) is linear. However, in order to build the system, we use an approximation (in implicit scheme) by Taylor expansion of non-linear term in equation (3.46). The procedure of linearization, when Δt goes toward infinity is similar to the Newton's method of finding roots of a non-linear equation,

$$f(x) = 0 \tag{3.58}$$

is given by,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{3.59}$$

thus the derivative is given by,

$$f'(x_n) = \frac{f(x_n)}{(x_{n+1} - x_n)} \tag{3.60}$$

Newtons' method converges to the exact solution with a quadratic slope when it starts from a suitable initial value. To obtain an exact solution at each step ("$_n$"), one needs to use a direct solver such as Gaussian elimination or Lower-Upper decomposition. However, here we deal with PDEs on a very large 3-dimensional grid which is very expensive. Alternatively, an iterative method can

be used by replacing the exact solution with an approximate one. This method is called Inexact-Newton-Iterative. [26].

In the inexact-Newton-iterative method, Newtons' method is used as outer loop to handle the non-linear part of systems. Then, the Newtons's loop continues until the Euclidean norm residual drops under the predefined tolerance,

$$\frac{\|\mathbf{W}_n\|}{\|\mathbf{W}_0\|} < tol_{res} \tag{3.61}$$

where tol_{res} is a user-defined tolerance value.

In addition, at each Newtons' iteration, the linear system, $(\mathbf{Ax} = \mathbf{b})$, is an inner loop. Krylov methods are the most popular iterative method to solve large sparse linear systems. The overall sequence is illustrated in Figure 3.9.

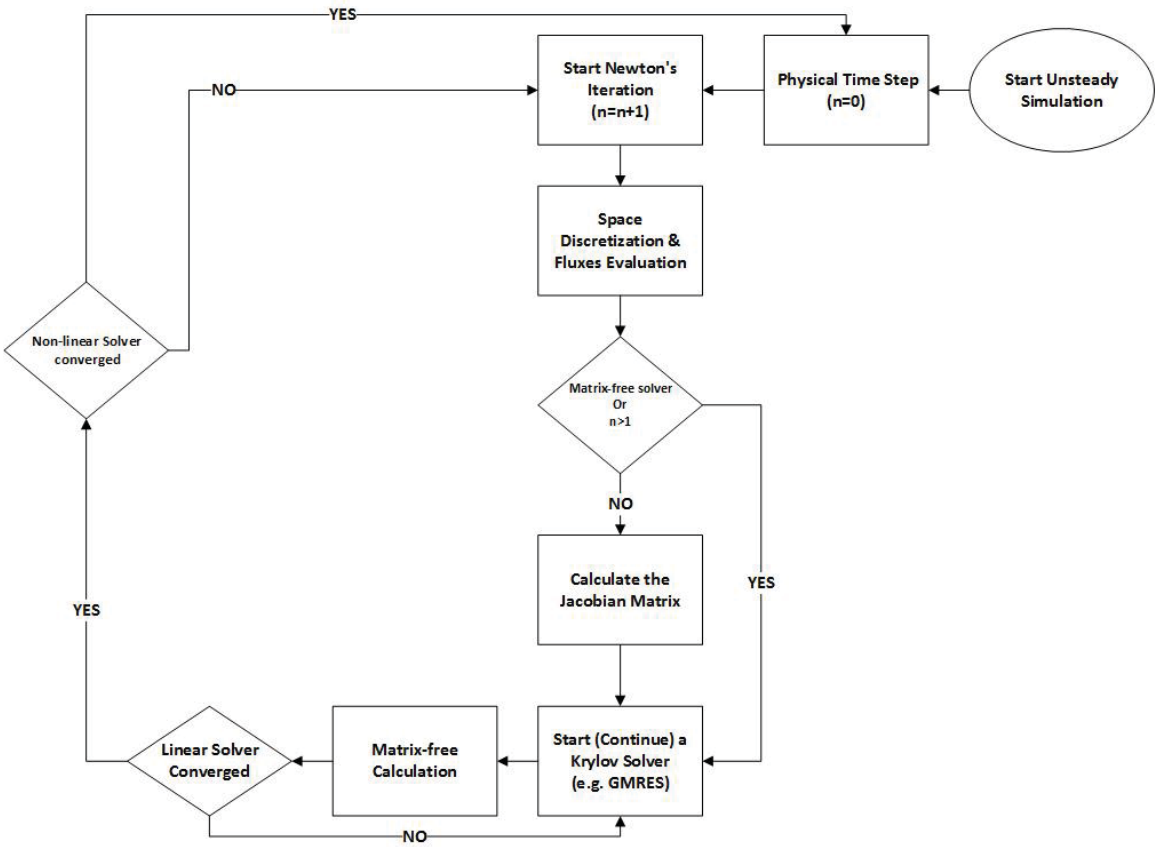


Figure 3.9: Flowchart of solving an unsteady flow by Inexat-Newton-Krylov method

3.9.1 Krylov based iterative methods

Krylov method, are iterative approaches to solve large sparse linear systems. The first methods, Conjugate Gradient (CG) models, were introduced as a direct solver [41], but later revised by Reid [92] to be used as an iterative solver. In such approaches the linear system is solved using a Krylov subspace, $\mathbf{K}_j$,

$$\mathbf{K}_j = span(\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{j-1}\mathbf{r}_0) \tag{3.62}$$

where j represents the number of subspaces, and $\mathbf{r}_0$ is the residual vector. Usually as j increases, it converges better to the solution. Larger subspace, however, increases the memory and computation costs.

A linear system, arising from the Navier-stokes equations is non-symmetric due to the hyperbolic nature of the N-S equations. Asymmetric matrix are only solved by methods that can handle non-symmetric Matrix such as Bi-Conjugate Gradient STABilized (BiCGSTAB) [12] and Generalized Minimum Residual Algorithm (GMRES) [94]. GMRES has been proven to be very efficient in turbulence flows and in the aerodynamics field [122]. It does not store the matrix coefficient separately, but only the Matrix-vector product is required at each inner loop. Herein, we repeat the restart version of the algorithm that is provided in [94],

Algorithm 1 GMRES(m)

1: *Start*: $Choose\ \mathbf{x}_0\ and\ compute\ \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0\ and\ \nu_1 = \mathbf{r}_0/||\mathbf{r}_0||$
2: *Iterate*: $For\ j = 1, 2, \ldots, m\ to\ calculate\ l_2 - orthogonal\ bases\ \mathbf{V}_m$
3: *Form the approximate Solution*: $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m$
4: *Restart*: *If convergence achieved then stop else* $\mathbf{x}_m = \mathbf{x}_0$ *and go to 2*

The value of $\mathbf{y}_m$ in step 3 of algorithm 1 is obtained by minimizing a function. This procedure includes an iterative inner loop. Interested readers may refer to reference [34] to find more details about implementation of the model. In order to improve the convergence, GMRES is combined with Preconditioning. The preconditioner role is to make a matrix well-conditioned, in order to improve its efficiency. The Jacobi preconditioner is one the most common one to GMRES method. It is very efficient in a Parallel code, although, it does not improve the convergence as well as more sophisticated method such Lower-Upper Symmetric Gauss-Seidel (LU-SGS). Consider the final

linear equation introduced in equation 3.57. We may write in the current form,

$$(\mathbf{D}+\mathbf{S})\Delta\mathbf{W}=\mathbf{R}, \tag{3.63}$$

where the change in fluxes ($\Delta\mathbf{W}=\mathbf{W}^{n+1}-\mathbf{W}^{n}$) and $\mathbf{D}$ and $\mathbf{S}$ are diagonal and off-diagonal part of the system matrix. The diagonal portion requires $(npoints \times neqns \times neqns)$ memory storage, while the off-diagonal needs a storage of $(2 \times nedges \times neqns \times neqns)$ [62, 63]. Jacobi preconditioner is a left-type one. It means that it is multiplied to both sides of equations,

$$(\mathbf{I}+\mathbf{D^{-1}S})\Delta\mathbf{W}=\mathbf{D}^{-1}\mathbf{R} \tag{3.64}$$

Since the matrix that is used in order to smooth the solution in Jacobian scheme only contains diagonal term, and corresponding to the node value and not its neighbors. Parallelization of this code can be easily performed. It should be noticed that the Jacobian Matrix has to be multiplied on the right hand side of the system. In addition, the restarted GMRES Algorithm need to be updated by multiplying the term $\mathbf{V}_m\mathbf{y}_m$ in the third step.

The last concern that rises regarding solving the linear system (3.57) is related to the derivative term ($\mathbf{J}=((\partial\mathbf{K}_i/\partial\mathbf{W}_i)|^n)$). Two approaches are available to solve this term: one is the direct approach that takes the derivative of fluxes directly, and build a Matrix. Jacobian of the convective and diffusive flux are shown in Appendix A. This method requires considerable mathematical effort. However, it is calculated only once at each time step. Second method is the so-called Matrix-free or Jacobian-free solvers [52], significantly reduce the complication of programming. Nevertheless one uncertainty is that they need to be updated at each iteration of the linear solver. Therefore, by increasing the size of mesh, it would be computationally expensive. Recall that in many Krylov based iterative solvers, such as GMRES (step 3 in Algorithm 1), the inner product of the Matrix-vector is required, not the Matrix itself. Therefore, the Matrix-vector $(\mathbf{Jv})$is approximated by,

$$\mathbf{Jv}\approx\frac{\boldsymbol{K}(\mathbf{W}+\epsilon\mathbf{v})-\boldsymbol{K}(\mathbf{W})}{\epsilon} \tag{3.65}$$

This approximation is derived by taking a first-order Taylor expansion about $\mathbf{W}$. The perturbation parameter, ϵ, is very small. There are various type of method to calculate ϵ [52]. Here we use the method introduced by Osusky [83].

$$\epsilon = \sqrt{\frac{N_u \delta}{\mathbf{v}^T \mathbf{v}}} \tag{3.66}$$

where, N_u denote the number of unknowns (in 3-D compressible flow $5 \times npoints$) and $\delta = 10^{-12}$. Higher order schemes are not required to calculate the fluxes, $(\boldsymbol{K}(\mathbf{W})$, in (3.65), since this only define the slope of convergence. We use the flux approximation introduced by Lou et al. [62], and revise the flux at the interface of node i and j,

$$\mathbf{F}_{ij} = \frac{1}{2}[^c\boldsymbol{F}_i + ^c\boldsymbol{F}_j] - \frac{1}{2}|\lambda|(\boldsymbol{W}_j - \boldsymbol{W}_i) \tag{3.67}$$

where $|\lambda|$ is given by,

$$|\lambda| = |\mathbf{V}_{ij}.\vec{n}_{ij}| + c_{ij} + \frac{\mu_{ij}}{\rho ij|\mathbf{x}_j - \mathbf{x}_i|} \tag{3.68}$$

where c_{ij}, $\mathbf{V}_{ij}$, and $\vec{n}_{ij}$ are the speed of sound, the velocity magnitude, and the normal at the interface of nodes i and j.

Chapter 4

Dynamic Mesh Methods

4.1 Dynamic Mesh Methods

The simulation of a deforming body such as a morphing blade requires a dynamic method that can preserve the quality of the mesh.
Before presenting appropriate dynamic mesh methods, a number of parameters are first introduced regarding generation of a valid co;/mputational domain either using an in-house code or commercial mesh generation software such as ICEM CFD, ANSYS Mesher, and etc. A valid mesh needs to pass two tests [47]: Topological and Geometrical tests. Topological tests are related to how the cells can be built from faces, or the faces built from edges and points. For example,

- A point cannot be repeated twice in a cell connectivity.
- A face can only be shared between a maximum of two cells.

More details about the topological criteria can be found in [47]. It is obvious that if a computational domain does not pass the topological test, it is impossible to solve on it even for a single time step. Therefore, these errors have to be diagnosed and fixed in pre-processing procedure.

Furthermore, checking the orthogonality is one of the main criteria in geometrical tests. Figure 4.1 shows two adjacent cells, where line $\overrightarrow{AB}$ is the line that connects the centroid of the cells and

N_s is the normal to the surface. $\overrightarrow{AB}.\vec{N}_s > 0$ is the orthogonality test.

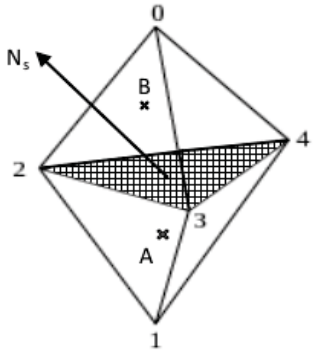


Figure 4.1: Orthogonality criteria

Passing both geometrical and topological tests result in a good quality mesh which is a pre-requisite for an accurate solution in Computational Fluid Dynamics (CFD). The geometrical test is related to the position of nodes inside the computational domain, in contrast to the topological test that depends on the mesh connectivity.
If a smoothing method is used to update the mesh, without re-meshing, the connectivity remains unchanged , while vertices are relocated in space. Therefore in smoothing methods, one only needs to be concerned with the geometrical test.

A prescribed profile usually represents a body surface that undergoes a deformation at each iteration. Then the interior cells should be adjusted, consequently, to the instantaneous position of boundary points. For aerodynamic studies, and specially in Large Eddy Simulation, the quality and size of mesh cells around a solid body play a significant role in the accuracy of the results. Therefore, preserving both quality and size of cells in the vicinity of a body should be considered carefully.

A smoothing approach sometimes is accompanied with re-meshing. In this approach, cells can be locally split or merged to improve the mesh quality, and they are very popular in multi-phase flow simulations. Since the connectivity is changed, the topological test is required. However, this method usually results in rebuilding the cell connectivity, and sometimes repartitioning the domain

is necessary for a parallel computing. Re-meshing by itself does not also guarantee preserving the mesh quality in the boundary layer.

In the current research, only smoothing technique are used without any re-meshing. This decision is based on simulations conducted with commercial software ANSYS FLUENT. The performed simulations showed that for an airfoil case undergoing deformation and re-meshing resulted in some negative effects on both computational time and mesh quality [107].

Two major approaches are widely used as smoothing methods: Spring method and Diffusion method. In the earlier approach, as it can be guessed from its name, the dynamic equations governed for a spring is employed to relocate the interior cells, while in the later a Laplace equation needs to be solved. In the following, the implementation of both methods in the in-house code is discussed in details.

Spring-base smoothing

In this method, it is assumed that each grid point follows Hooke's law. Therefore at equilibrium, the net force due to the springs (edges) connected to each vertex is zero (figure 4.2). The resultant force based on linear spring correlation is given by,

$$F_i = \sum_{j}^{n_i} k_{ij}(\Delta\vec{x}_i - \Delta\vec{x}_j), \tag{4.1}$$

where k_{ij} represents the spring stiffness and n_i is the number of connected edges to vertex i. The vertex displacements, $\Delta\vec{x}_i$ and $\Delta\vec{x}_j$, correspond to the points i and j, respectively. Since at equilibrium F_i is zero, the displacement of each vertex can be calculated from,

$$\Delta\vec{x}_i = \frac{\sum_j^{n_i} k_{ij}\Delta\vec{x}_j}{\sum_j^{n_i} k_{ij}}. \tag{4.2}$$

A corrector-predictor method is used to evaluate the displacement at each vertex [7]. Then, a linear extrapolation is utilized,

$$\Delta\vec{x}_i = 2\Delta\vec{x}_i^{n} - \Delta\vec{x}_i^{n-1}. \tag{4.3}$$

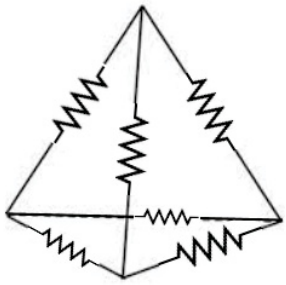


Figure 4.2: Spring analogy for a tetrahedral element

Finally, an iterative method is used to calculate the new position of the vertices. This iterative method continues until either the convergence criteria is meet or it reaches to the pre-defined maximum number of iterations.

Usually by approaching the region close to a body, for aerodynamic applications, the grid size typically reduces in order to capture small vortices and eddies. In addition, it is important to preserve the normal distance of grid points away from the body surface in the boundary layer region. In a linear spring analogy, the most important and influential parameter is the way the stiffness coefficient is evaluated, where the most common method used in many commercial software such as ANSYS FLUENT, is an inverse correlation between segment length and stiffness coefficient.

$$k_{ij} = \frac{1}{d_{ij}}, \tag{4.4}$$

where d_{ij} is the distance between point i and j,

$$d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2}. \tag{4.5}$$

Some problems required a uniform grid, while others, such as boundary layer flows, contain cells with the high aspect ratio where preserving the orthogonality of the cells close to the body is very important. However, equation (4.4) limits the correlation between the edge size and the stiffness with the reciprocal function. Blom [14] showed that adding a power coefficient may improve the accuracy of the base model introduced by Batina [7],

$$k_{ij} = \frac{1}{d_{ij}^{\kappa}}, \tag{4.6}$$

where a reasonable range for κ is between 0.5 to 3.0, based on experience from tests.

Adding the power coefficient enables the adjustment of the stiffness coefficient corresponding to the grid objective. However, there are two modes of failure that come along with the linear spring analogy are not alleviated. These modes are shown in Figure 4.3 . The first mode of failure is the coincident point error that happens when an edge length goes to zero. This occurs when an edge is stretched or shorten, and the inverse correlation between stiffness and the segment length causes to it continues until it makes the computational grid invalid in the next iterations. Some papers suggest an approach to avoid the coincident point by using a non-linear spring or exponential law spring [47]. Although, the proposed corrections may resolve some grid difficulties, they impose a considerable computational cost to the solver.

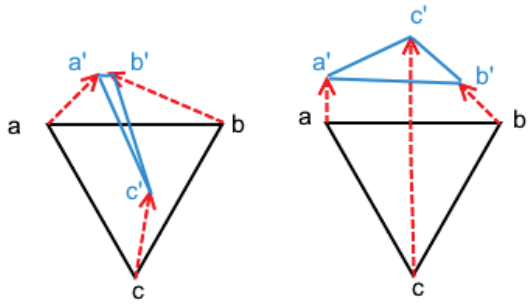


Figure 4.3: Spring analogy errors

Herein, in order to counter the edge expansion or contraction, a new method is introduced to modify the edge length, with negligible extra computational burden. It prevents any coincident point failure in the domain. Introducing,

$$k_{ij} = \frac{1}{(Cd_{ij})^{\kappa}} \tag{4.7}$$

where C is given by,

$$C = \begin{cases} max(1.0, 1.0/R) & R \leq C_{ex}, \\ 1.0 & C_{ex} < R < 1.0/C_{ex}, \\ min(1.0, 1.0/R) & R \geq 1.0/C_{ex}, \end{cases} \tag{4.8}$$

and $R = d_{ij}^{n}/d_{ij}^{n-1}$ is the ratio of segment length between two sequential time. C_{ex} is a user-defined parameter that define the limit which applies the correction coefficient C. It can take a value of $0 < C_{ex} \leq 1$. The introduced model eases edge expansion and contraction.

The second mode of failure corresponding to the linear spring method, shown in Figure 4.3, is called triangle flip or snap-through of a cell. This results in interpenetration of cell into its neighbour cell when a vertex crosses over an edge in 2D or a face in 3D domain. Fahat et al. [25,31] suggested to replace the linear spring with a torsional spring at each vertex in 2D and 3D to control the angle between edges (Figure 4.4). They concluded that a torsional spring improves the robustness of the spring analogy, both to avoid merging of vertices and neighbour cell interpenetration. However, the torsional spring increases the computational costs. Also, the non-linear system of equations also may cause to some convergence problems.

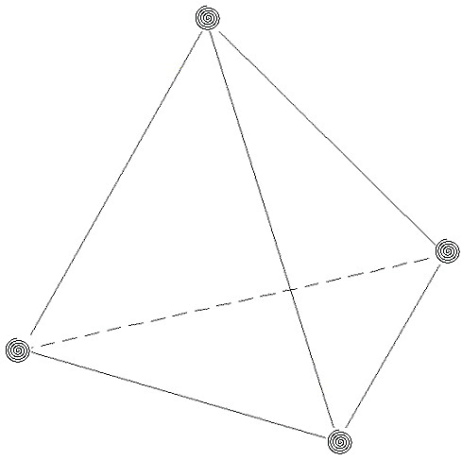


Figure 4.4: Torsional spring analogy on a tetrahedral cell for a 3-D mesh

Laplace-based smoothing

For the diffusion method, the Laplace equation has to be solved,

$$\begin{cases} \nabla.(\gamma \nabla \mathbf{u}) = 0 & on\ \Omega \\ u = f & on\ \delta\Omega \end{cases} \tag{4.9}$$

where, γ is the diffusion parameter defined as,

$$\gamma = \frac{1}{d^n} \tag{4.10}$$

Two approches can be used to control the diffusion coefficient: volume-based and distance-based. In the volume-based approach, d is calculated based on the cell volume, while the in distance-based approch, d is evaluated based on distance of each vertex to the body. In both models, the value of diffusion increases when approaching the body surface.

In order to solve the Laplace equation, a Galerkin finite element method is implemented. Then, the weak form of equation (4.9) is built,

$$a(u, v) = \int_{Tet_k} \gamma \nabla u \nabla v = 0, \tag{4.11}$$

where u is the grid velocity and $v = \phi_i$, it results in a linear system,

$$KU = 0, \tag{4.12}$$

where K is the Stiffness Matrix, and the same linear base function ϕ over tetrahedral element that was introduced that is introduced in equation (3.11) is used to create a second order accuracy solution,

$$\mathbf{u}(x, y, z) = \sum_{i=1}^{4} \mathbf{u}_i \phi_i. \tag{4.13}$$

Therefore, the stiffness Matrix array are given by,

$$K_{ij} = \sum_{k=1}^{N} \int_{Tet_k} \gamma \nabla \phi_i \nabla \phi_j \tag{4.14}$$

Since the base function is linear,

$$\nabla\phi_i = \begin{bmatrix} b_i \\ c_i \\ d_i \end{bmatrix}. \tag{4.15}$$

Then, the integral equation (4.14) is rewritten,

$$\int_{Tet_k} \gamma \nabla\phi_i \nabla\phi_j = \nabla\phi_i \nabla\phi_j \int_{Tet_k} \gamma. \tag{4.16}$$

The integral is approximated by the one-point rule,

$$\int_{Tet_k} \gamma = V * \bar{\gamma}, \tag{4.17}$$

where V is cell volume and $\bar{\gamma}$ is averaged over each cell,

$$\bar{\gamma} = \frac{\gamma_1 + \gamma_2 + \gamma_3 + \gamma_4}{4}. \tag{4.18}$$

The first term on the right hand side of equation (4.16) is given by,

$$\nabla\phi_i \nabla\phi_j = \begin{bmatrix} b_1 & c_1 & d_1 \\ b_2 & c_2 & d_2 \\ b_3 & c_3 & d_3 \\ b_4 & c_4 & d_4 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{bmatrix} \tag{4.19}$$

Before solving the system, one needs to deal with inhomogeneous Dirichlet boundary condition. Then, the following term is added to the right hand side equation (4.11),

$$F_i = -\sum_{k=1}^{N} \int_{Tet_k} \gamma \nabla G \nabla\phi_i, \tag{4.20}$$

where G is the boundary condition,

$$\nabla G = \sum_{i=1}^{4} f_i \nabla\phi_i, \tag{4.21}$$

where f_i is the vertex velocity on the boundary surface, and it is zero for the interior nodes. The

integral on the right hand side of equation (4.20) can be expanded to,

$$\int_{Tet_k} \gamma \nabla G \nabla \phi_i = \bar{\gamma} * V * \begin{bmatrix} b_1 & c_1 & d_1 \\ b_2 & c_2 & d_2 \\ b_3 & c_3 & d_3 \\ b_4 & c_4 & d_4 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \\ d_1 & d_2 & d_3 & d_4 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{4.22}$$

Finally, the GMRES algorithm is used to solve this linear system.

4.2 Arbitrary Lagrangian-Eulerian Equation

In the previous section, a few strategies were described to update the interior points when some boundaries underwent an arbitrary deformation. In fluid mechanics, two major approaches are used to analyse fluid flows [73]: Eulerian and Lagrangian approaches. In the former approach, a fixed reference frame is considered and the flow kinematics is calculated at each time and position with respect to the reference frame, whereas the Lagrangian approach tracks fluid particles. From a computational point of view, computational mesh is fixed in the Eulerian approach, and fluid velocity is evaluated relative to this grid. In contrast, in the Lagrangian approach, mesh grid moves with the individual particles. Equation 3.1 is derived based on the Eulerian approach in analyzing fluid dynamics. However, it is clear that the governing system of equations may not remain the same while grid points move. Donea et al. [28] proposed using Arbitrary Lagrangian-Eulerian (ALE) method to take advantage of both aforementioned approaches and minimizing their negative aspects. They clearly depicted differences between three different approaches in Figure 4.5 where fluid particles and computational nodes in one-dimensional space are shown. As it is seen, in the Lagrangian model, particles and grid points are displaced identically, while in the Eulerian method, grid points do not move in time. Finally, in ALE, nodes velocity is neither zero (Eulerian) nor identical to particle velocity (Lagrangian), but have a value in between.

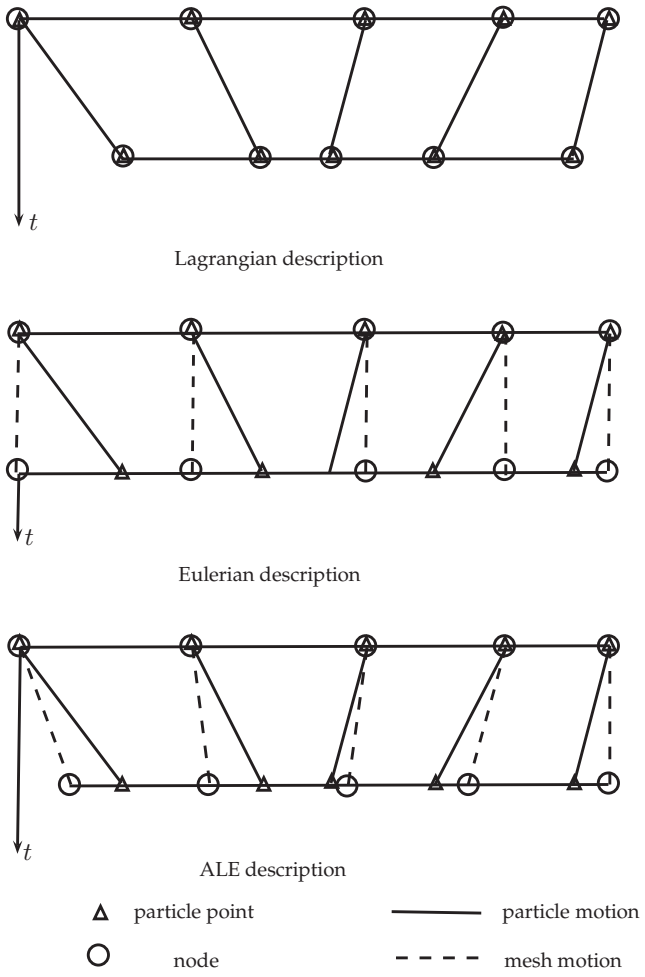


Figure 4.5: Difference between Eulerian, Lagrangian and ALE approach [28]

The Arbitrary Lagrangian Eulerian technique forms a system of governing equation that takes into account these grid points movement. Lesoinne and Farhat [59, 60] developed a first order time accurate method based on ALE to simulate a fluid problem that undergoes a movement on unstructured grid in two and three-dimensional space. They showed that preserving Conservative Geometric Law (GCL) is crucial to solve a time-accurate dynamic mesh. A dynamic mesh approach is called GCL if the sweep area (volume) by edges (faces) is equal to the variation in area (volume) of the cell. This condition can be written as,

$$vol(I, t^{n+1}) - vol(I, t^n) = \int_{t^n}^{t^{n+1}} \int_{\partial C(I)} \dot{x}.\vec{n} dA dt \tag{4.23}$$

Koobus and Farhat [53] derived higher accurate solution of two and three-dimensional viscous

fluxes. They [54] extended their research to the Navier-Stokes equations by using higher order time-accurate methods to solve dynamic meshes. Herein, a brief description of their method is presented, which is also implemented in the current in-house code.

In a Cartesian system of coordinate points, x is used to show the coordinate associated with time t. The reference coordinate are shown with ξ and it is measured at time τ. Base on this definition, at any arbitrary time, a mapping function can be defined that correlates the current time to the reference frame. Then the ALE conservative form of the Navier-Stokes equations is given by,

$$\begin{aligned} \frac{\partial \mathbf{JW}}{\partial t}|_{\xi} + J\nabla.({}^{c}\mathbf{F}(\mathbf{W},\dot{x}) + {}^{v}\mathbf{F}(\mathbf{W})) &= \mathbf{S}(\mathbf{W}) \\ {}^{c}\mathbf{F}(\mathbf{W},\dot{x}) = {}^{c}\mathbf{F}(\mathbf{W},\dot{x}) - \dot{x}\mathbf{W}&, \end{aligned} \tag{4.24}$$

where $\mathbf{J} = det(dx/d\xi)$ is the derivative of the coordinate location associated with the flow configuration at the current time t, with respect to a reference configuration at time τ. The notation $\dot{x} = (\partial x/\partial\tau)|_{\xi}$ is a time derivative of the grid configuration.

Discretization of the convective, diffusive and the source term can be performed similarly as described in section 3.2.1 to section 3.2.3, except that the convective flux is modified. Then, the final equation at each vertex can be written as,

$$\frac{d}{dt}\int_{C(I,t)} \mathbf{W} dV = R_C(\mathbf{W}, X, \dot{X}) + R_V(\mathbf{W}, X) \tag{4.25}$$

Note that the Source term is neglected in equation 4.25. $R_C(\mathbf{W}_i, X, \dot{X})$ and $R_V(\mathbf{W}_i, X)$ are distributions of convective and diffusion terms on the right hand side. One may rewrite equation (4.25) as,

$$\frac{d(vol(C_I,t)\mathbf{W}_I)}{dt} = R_C(\mathbf{W}, X, \dot{X}) + R_V(\mathbf{W}, X) \tag{4.26}$$

where W_I is the averaged conservative variable at cell $C(I)$ at time t. The notation $vol(C_I, t)$ is its corresponds to the cell volume. Therefore equation (4.25) can be written as,

$$vol(C_I, t^{n+1})\mathbf{W}_I^{n+1} - vol(C_I, t^n)\mathbf{W}_I^n = \int_{t^n}^{t^{n+1}} R_C(\mathbf{W}, X, \dot{X})dt + \int_{t^n}^{t^{n+1}} R_V(\mathbf{W}, X)dt \quad (4.27)$$

Two important questions have to be answered while a solution for an unsteady dynamic problem is considered: How is the metric parameter of mesh calculated? How is the integral on the right hand side of equation (4.27) calculated? Answers to these questions are very important to guarantee the conservative geometric law. The metric parameter includes normal of the surfaces, cell volume, and etc. For a stationary case, these parameter only need to be calculated once at the beginning of solution, and they remain the same during the rest of the simulation. On the other hand, for a dynamic mesh, it is important that these value are calculate at time t^n, t^{n+1}, or a combination of the mesh configuration at these times. In order to calculate the integral on the right hand side, the mid point approach, two-point approach or a higher order method can be used. Choosing the integral method directly affects the accuracy of the solution.

4.2.1 Integral time diffusion term

Koobus and Farhat [53] showed that the following numerical methods may be used to calculate the viscous term integral, while GCL is satisfied on a tetrahedral grid with a piece-wise linear finite element method. One point Integral can be used,

$$\begin{aligned} &\int_{t^n}^{t^{n+1}} R_V(\mathbf{W}, X)dt = \Delta t R_V(\mathbf{W^k}, X^k) + O(\Delta t^2) \\ &\qquad k = n \qquad\qquad \text{explicit method} \\ &\qquad k = n+1 \qquad\qquad \text{implicit method} \end{aligned} \quad (4.28)$$

As it is seen, using the one-point integral results in a second order time-accurate solution. The time that fluxes are calculated depends upon using an explicit or implicit method. The viscous integral can be calculated from the following equation, if a mid-point numerical method to calculate the integral is considered,

$$
\begin{aligned}
&\int_{t^n}^{t^{n+1}} R_V(\mathbf{W}, X)dt = \Delta t R_V(\mathbf{W^k}, X^{n+\frac{1}{2}}) + O(\Delta t^3) \\
&k = n \qquad \text{explicit method} \\
&k = n + \frac{1}{2} \quad \text{implicit method} \\
&\mathbf{W}^{n+\frac{1}{2}} = \frac{\mathbf{W}^{n+1} - \mathbf{W}^n}{2} \\
&X^{n+\frac{1}{2}} = \frac{X^{n+1} - X^n}{2}
\end{aligned}
\tag{4.29}
$$

This mid-point method results in second order accuracy. Finally, the two-point integral is given by,

$$
\begin{aligned}
&\int_{t^n}^{t^{n+1}} R_V(\mathbf{W}, X)dt = \frac{\Delta t}{2}[R_V(\mathbf{W}^{k1}, X^{m1}) + R_V(\mathbf{W}^{k2}, X^{m2})] + O(\Delta t^5). \\
&m1 = n + \frac{1}{2} - \frac{1}{2\sqrt{3}} \\
&m2 = n + \frac{1}{2} - \frac{1}{2\sqrt{3}} \\
&k1 = n \quad \text{explicit method} \\
&k2 = n \quad \text{explicit method} \\
&k1 = m1 \quad \text{implicit method} \\
&k2 = m2 \quad \text{implicit method} \\
&\mathbf{W}^{n+\beta} = \beta \mathbf{W}^{n+1} + (1-\beta)\mathbf{W}^n \\
&X^{n+\beta} = \beta X^{n+1} + (1-\beta)X^n
\end{aligned}
\tag{4.30}
$$

Then a fourth-order time accurate solution is achieved with the two-point integral defined in equation 4.30. Note that all three integrals satisfy the conservative geometric law, though the solution accuracy varies from one to another.

4.2.2 Integral time convection term

To evaluate the integral of the convection term, one substitutes the ALE convective term from equation (4.24) into equation (3.21) and writes,

$$\int_{t^n}^{t^{n+1}} \int_{C(i)} {}^{c}\boldsymbol{F}.\vec{n}_{\partial C(i)} dS = \int_{t^n}^{t^{n+1}} \left[\sum_{J \in k(i)} \left[\int_{\partial C(ij)} {}^{c}\mathbf{F}(W_h).\vec{n}_{\partial C(ij)} ds - \int_{\partial C(ij)} W_h \dot{\vec{x}}.\vec{n}_{\partial C(ij)} ds \right] \right] dt \tag{4.31}$$

Then one can follow one of two strategies to calculate a time-accurate integral on the convection term. In the first approach, the normal faces and grid speed are calculated in a way that a GCL is satisfied. Nkonga and Guillard [81] proposed a method to calculate the metric parameters on 3-dimensional unstructured grids along with a moving boundary. Herein , their method is summarized.

If the convective fluxes are constant on the interface of two neighbouring dual-cell, the following two have to be evaluated,

$$\begin{aligned} \vec{\eta}_{ij} &= \frac{1}{\Delta t} \int_{t^n}^{t^{n+1}} \int_{\partial C(ij)} \vec{n}_{\partial C(ij)} dsdt, \\ \vec{\sigma}_{ij} &= \frac{1}{\Delta t} \int_{t^n}^{t^{n+1}} \int_{\partial C(ij)} \dot{\vec{x}}.\vec{n}_{\partial C(ij)} dsdt. \end{aligned} \tag{4.32}$$

Before discussing these integral furthermore, a short summary of how to calculate the geometric parameters on a tetrahedral dual-element are presented.

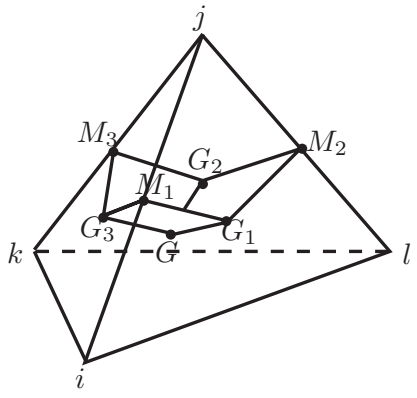


Figure 4.6: intersection between two adjacent dual-cell

In Figure 4.6, the details of a dual cell on a tetrahedral grid is is illustrated. The points M are the middle point of each edge,

$$\mathbf{X}_{M_1} = \frac{1}{2}(\mathbf{X}_i + \mathbf{X}_j); \quad \mathbf{X}_{M_2} = \frac{1}{2}(\mathbf{X}_j + \mathbf{X}_l); \quad \mathbf{X}_{M_3} = \frac{1}{2}(\mathbf{X}_j + \mathbf{X}_k) \tag{4.33}$$

The G points represents the center of gravity of a tetrahedral cell and its faces,

$$\begin{aligned} \mathbf{X}_{G_1} &= \frac{1}{3}(\mathbf{X}_i + \mathbf{X}_j + \mathbf{X}_l); & \mathbf{X}_{G_2} &= \frac{1}{3}(\mathbf{X}_j + \mathbf{X}_k + \mathbf{X}_l); \\ X_{G_3} &= \frac{1}{3}(\mathbf{X}_i + \mathbf{X}_j + \mathbf{X}_k); & \mathbf{X}_G &= \frac{1}{4}(\mathbf{X}_i + \mathbf{X}_j + \mathbf{X}_k + \mathbf{X}_l) \end{aligned} \tag{4.34}$$

Common face between vertices i and j inside element E is a planar quad which is built by $(M_1 - G_1 - G - G_3)$, and is shown by $(\partial C_{ij} \cap E)$. Finding the normal to this face, as it is shown in equation (3.22), is part of the solution to evaluate the contribution of the convective term. Since the normal surface on each individual quad is constant then,

$$\vec{\nu}_{(ij \cap E)} = \int_{(\partial C(ij) \cap E)} \vec{n}_{(\partial C(ij) \cap E)} ds = \vec{n}_{(\partial C(ij) \cap E)} S_{(\partial C(ij) \cap E)} \tag{4.35}$$

The normal to a plane is perpendicular to two edges inside that plane, therefore,

$$\vec{n}_{(\partial C(ij) \cap E)} = \frac{G_1\vec{M}_1 \times G_3\vec{M}_1}{\|G_1\vec{M}_1 \times G_3\vec{M}_1\|} = \frac{\vec{G_3G} \times \vec{G_1G}}{\|\vec{G_3G} \times \vec{G_1G}\|} \tag{4.36}$$

In order to calculate $S_{(\partial C(ij) \cap E)}$, which is the area of the common surface between node i and j on tetrahedral element E, we can split it into two planar triangles (Figure 4.7). Therefore, the total area is the summation of the area of two triangles $T_1 : (M_1 - G_1 - G_3)$ and $T_2 : (G_1 - G - G_3)$. It can be shown that based on the relations that are defined in equations (4.33) and (4.34), area $S_1 = (1/2)\|G_1\vec{M}_1 \times G_3\vec{M}_1\|$ is twice of $S_2 = (1/2)\|\vec{G_3G} \times \vec{G_1G}\|$. Therefore, equation (4.36) is rewritten,

$$\begin{aligned} \vec{\nu}_{(ij \cap E)} &= \frac{3}{2} S_1 \vec{n}_{(\partial C(ij) \cap E)} & &= \frac{3}{2}(G_1\vec{M}_1 \times G_3\vec{M}_1) \\ &= 3S_2 \vec{n}_{(\partial C(ij) \cap E)} & &= 3(\vec{GG}_1 \times \vec{GG}_3) \end{aligned} \tag{4.37}$$

A stationary grid has already been discussed so far. In the case of a dynamic mesh, the normal surfaces varies in time, then it is important to take the time integral to make sure the GCL is satisfied. Vertices can move with different velocity, however, it is assumed their velocity is constant between

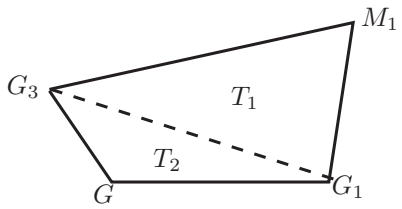


Figure 4.7: piece-wise linear velocity displacement

time t^n and t^{n+1}. It is depicted in Figure 4.8 for a face.

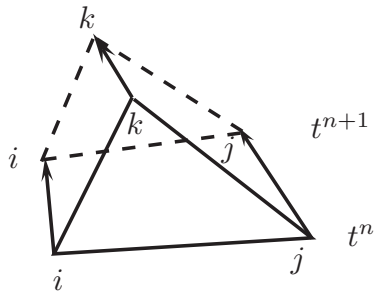


Figure 4.8: piece-wise linear velocity displacement.

Nkonga and Guillard [81] showed that the time-integral is given by,

$$\begin{aligned}
\vec{\eta}_{(ij\cap E)} &= \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \int_{(\partial C(ij)\cap E)} \vec{n}_{(\partial C(ij)\cap E)} ds dt = \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \vec{\nu}_{(ij\cap E)} dt \\
&= \frac{1}{3}(\vec{\nu}^{n+1}_{(ij\cap E)} + \vec{\nu}^{n^\star}_{(ij\cap E)} + \vec{\nu}^{n}_{(ij\cap E)}) \\
\vec{\nu}^{n^\star}_{(ij\cap E)} &= 3[(\mathbf{X}^{n+1}_{G_1} - \mathbf{X}^{n+1}_{G}) \times (\mathbf{X}^{n}_{G_3} - \mathbf{X}^{n}_{G}) + (\mathbf{X}^{n}_{G_1} - \mathbf{X}^{n}_{G}) \times (\mathbf{X}^{n+1}_{G_3} - \mathbf{X}^{n+1}_{G})]
\end{aligned} \tag{4.38}$$

Equation (4.38) represents the normal surface vector between point i and j inside element E. Therefore to find the solution to the first equality in equation (4.32), all the supporting elements that have ij as their edge have to be included,

$$\vec{\eta}_{ij} = \sum_{E\in k(ij)} \vec{\eta}_{(ij\cap E)} \tag{4.39}$$

where $k(ij)$ are the supporting elements.

Now, the second equality in equation (4.32) is solved. The normal velocity for a triangular face

(Figure 4.8), under the assumption that the node velocity does not change between time t^n and t^{n+1}, is given by,

$$\vec{\sigma}_{ij} = \frac{1}{\Delta t \|\vec{\eta_T}\|}\Big[\vec{\dot{x}}_G \int_{t^n}^{t^{n+1}} \int\limits_{\partial C(ij)} \vec{n}_{\partial C(ij)} ds dt\Big] \quad = \frac{\vec{\dot{x}}_G.\vec{\eta_T}}{\Delta t \|\eta_T\|} \tag{4.40}$$

$\vec{\eta_T}$ is the normal face vector; $\vec{\dot{x}}_G$ is velocity of gravity center of facet which can be calculated by averaging the value over its vertices.

$$\vec{\dot{x}}_G = \frac{\vec{\dot{x}}_i + \vec{\dot{x}}_j + \vec{\dot{x}}_k}{3} \tag{4.41}$$

Then, the normal velocity at the interface between nodes i and j inside the element is given by,

$$\vec{\sigma}_{(ij\cap E)} = \frac{1}{\Delta t} \int_{t^n}^{t^{n+1}} \int\limits_{(\partial C(ij)\cap E)} \vec{\dot{x}}_G.\vec{n}_{(\partial C(ij)\cap E)} ds dt \tag{4.42}$$

As it is seen earlier, the intersection may be split into two triangular faces T_1 and T_2,

$$\begin{aligned}
\vec{\sigma}_{(ij\cap E)} &= \frac{1}{\Delta t \|\vec{\eta_{ij}}\|} \int_{t^n}^{t^{n+1}} \big(S_1 \vec{\dot{x}}_{G(T_1)}.\vec{n}_{(\partial C(ij)\cap E)} + S_2 \vec{\dot{x}}_{G(T_2)}.\vec{n}_{(\partial C(ij)\cap E)}\big) dt \\
&= \frac{1}{\Delta t \|\vec{\eta_{ij}}\|} \int_{t^n}^{t^{n+1}} \big(2S_2 \vec{\dot{x}}_{G(T_1)}.\vec{n}_{(\partial C(ij)\cap E)} + S_2 \vec{\dot{x}}_{G(T_2)}.\vec{n}_{(\partial C(ij)\cap E)}\big) dt \\
&= \frac{(2\vec{\dot{x}}_{G(T_1)} + \vec{\dot{x}}_{G(T_2)})}{\Delta t \|\vec{\eta_{ij}}\|}. \int_{t^n}^{t^{n+1}} \big(S_1 \vec{n}_{(\partial C(ij)\cap E)}\big) dt
\end{aligned} \tag{4.43}$$

Combining this relation with equations (4.37) and (4.38) results in,

$$\vec{\sigma}_{(ij\cap E)} = \frac{(2\vec{\dot{x}}_{G(T_1)} + \vec{\dot{x}}_{G(T_2)})}{\Delta t \|\vec{\eta_{ij}}\|}.\vec{\eta}_{(ij\cap E)} = \frac{\vec{\dot{x}}_{(ij\cap E)}.\vec{\eta}_{(ij\cap E)}}{\|\vec{\eta_{ij}}\|}, \tag{4.44}$$

where $\vec{\dot{x}}_{(ij\cap E)}$ is the average normal velocity inside element E. Considering equation (4.41), it leads to,

$$\vec{\dot{x}}_{(ij\cap E)} = \frac{(2\vec{\dot{x}}_{G(T_1)} + \vec{\dot{x}}_{G(T_2)})}{3} \quad = \frac{1}{36}(13\vec{\dot{x}}_i + 13\vec{\dot{x}}_j + 5\vec{\dot{x}}_k + 5\vec{\dot{x}}_l) \tag{4.45}$$

The contribution from all the support elements that have ij as their edge has to be summed to calculate the mean value of the normal velocity:

$$\vec{\sigma_{ij}} = \sum_{E \in k(ij)} \vec{\sigma}_{(ij \cap E)} = \frac{1}{\|\vec{\eta_{ij}}\|} \sum_{E \in k(ij)} \vec{\dot{x}}_{(ij \cap E)} . \vec{\eta}_{(ij \cap E)} \tag{4.46}$$

Therefore, the convective discretized term defined in equation (3.21) for the case of a dynamic mesh changed to,

$$\int_{t^n}^{t^{n+1}} \int_{C(ij)} {}^{c}\boldsymbol{F} . \vec{n}_{\partial C(ij)} dS = \mathbf{F}_{ij} . \vec{\eta_{ij}} - \vec{\sigma_{ij}} \|\vec{\eta_{ij}}\| \partial W_{ij} \tag{4.47}$$

where $\mathbf{F}_{ij}$ can be obtained from any method that was previously described such as Roe, Roe-MUSCL, and etc.

As proposed by Nkonga and Guillard [81], taking the mean value of the normal of the surface and velocity results in second order accuracy both in time and space.

Koobus and Farhat [54] proposed another approach that also results in second order accuracy, and also preserve GCL. They suggested to calculate the time-integral of the convective flux by,

$$\begin{aligned} \int_{t^n}^{t^{n+1}} R_C(\mathbf{W}, \mathbf{X}, \dot{X}) dt &= \Delta t^n \sum_{k=1}^{4} \omega_k R_C(\mathbf{W}^{n+1}, \mathbf{X}^{mk}, \dot{\mathbf{X}}^{mk}) dt \\ \mathbf{X}^{mk} &= \zeta_{n+1}^{mk} \mathbf{X}^{n+1} + \zeta_n^{mk} \mathbf{X}^n + (1 - \zeta_n^{mk} - \zeta_{n+1}^{mk}) \mathbf{X}^{n-1} \\ \dot{\mathbf{X}}^{mk} &= \theta_{n+1}^{mk} \mathbf{X}^{n+1} + \theta_n^{mk} \mathbf{X}^n + (1 - \theta_n^{mk} - \theta_{n+1}^{mk}) \mathbf{X}^{n-1} \end{aligned} \tag{4.48}$$

Coefficients in equation 4.48 are given by,

$$
\begin{cases}
\delta_1 = \frac{1}{2}(1 - \frac{1}{\sqrt{3}}), \quad \delta_1 = \frac{1}{2}(1 + \frac{1}{\sqrt{3}}) \\
\zeta_n^{m1} = \delta_2, \quad \zeta_n^{m2} = \delta_1, \quad \zeta_n^{m3} = \delta_1, \quad \zeta_n^{m4} = \delta_2, \\
\zeta_{n+1}^{m1} = \delta_1, \quad \zeta_{n+1}^{m2} = \delta_2, \quad \zeta_{n+1}^{m3} = 0, \quad \zeta_{n+1}^{m4} = 0, \\
\theta_n^{m1} = -\frac{1}{\Delta t^n}, \quad \theta_n^{m2} = -\frac{1}{\Delta t^n}, \quad \theta_n^{m3} = \frac{1}{\Delta t^{n-1}}, \quad \theta_n^{m4} = \frac{1}{\Delta t^{n-1}}, \\
\theta_{n+1}^{m1} = \frac{1}{\Delta t^n}, \quad \theta_{n+1}^{m2} = \frac{1}{\Delta t^n}, \quad \theta_{n+1}^{m3} = 0, \quad \theta_{n+1}^{m4} = 0, \\
\omega_1 = \frac{\alpha_{n+1}}{2}, \quad \omega_2 = \frac{\alpha_{n+1}}{2}, \quad \omega_3 = -\frac{\alpha_{n-1}}{2\tau}, \quad \omega_4 = -\frac{\alpha_{n-1}}{2\tau}, \\
\alpha_{n+1} = \frac{1+2\tau}{1+\tau}, \quad \alpha_{n+1} = \frac{\tau^2}{1+\tau}, \quad \tau = \frac{\Delta t^n}{\Delta t^{n-1}}
\end{cases}
\tag{4.49}
$$

Chapter 5

Convergence Study and Deformation Schemes Analysis

In this chapter, the investigate of the robustness of linear solver is presented followed by, a comparison of the the Jacobian-free and Jacobian-based models. Afterwards, the effects of different parameters on the convergence of system are compared for a bump case in an transient Euler flow. The Matrix-free solver and the Jacobian-based methods are compared on a subsonic bump flow. The robustness of PETSc solver and the current in-house code solver are analyzed for transonic flow over NACA0012. Also, the solver robustness of the Navier-Stokes equation is analyzed for a laminar flow in Pipe. Finally, the effectiveness of the deformation schemes are investigated.

5.1 Transonic Circular Arc Bump

A convergence study is performed on a bump with the boundary conditions and grid shown in Figure 5.1 and 5.2, respectively. Periodic condition are considered in the spanwise direction. The Bumps' height is $h = 0.1(m)$. The Compressible Euler equations are used to simulate the flow. A second order Roe-MUSCL with the Van-Albada limiter for space discretization and an implicit scheme with second order time accuracy are chosen. The GMRES solver is used to solve the linear system, which iterates until the relative residual inside the linear solver falls below $(O(10^{-5}))$. It is worth mentioning that the linear solver residual convergence limit may be chosen to be much larger

$(O(10^{-3})$ as it is a steady state solution. The matrix Jacobian is calculated by taking derivatives of the convective fluxes. The mesh includes 299880 points and 1708200 tetrahedral cells, where the aspect ratio of cells near the bump surface is unity in both normal and spanwise direction. The domain is partitioned into 48 subdomains with METIS [50].

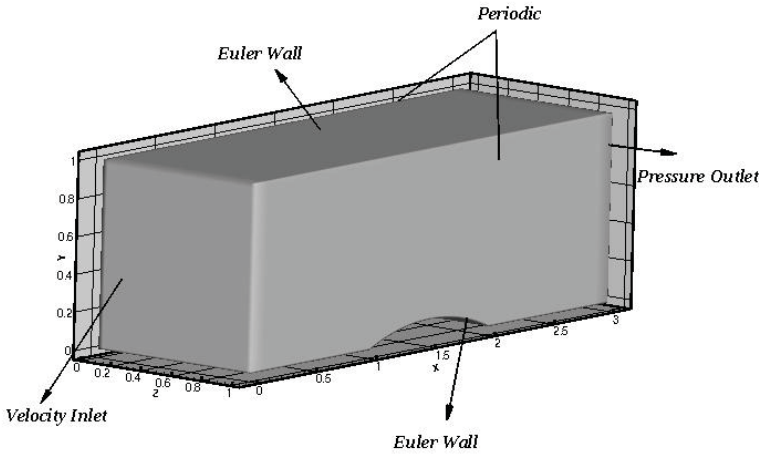


Figure 5.1: Bump Geometry and Boundary Condition.

Inlet conditions are given in Table 5.1. When the flow approaches the bump, it accelerates until a shock appears and flow returns to a subsonic regime.

Table 5.1: Inlet Condition and Geometry Property

Parameters	Mach Number	Bump Height
Value	0.675	10 %

Figure 5.3 shows the entropy contours. Entropy is constant all through the domain, except after the shock. At the shock, a jump is observed, and entropy increases closer to the bump's surface. The shock happens at $(x = 1.72)$, which is consistent with those in reference [27, 32]. The Mach number contours are shown in Figure 5.4.

Herein, the study of the convergence of the system and the effect of number of Newtons' outer loop, non-linear iterations (nbnewton), are presented. As it is seen all the conservative variables

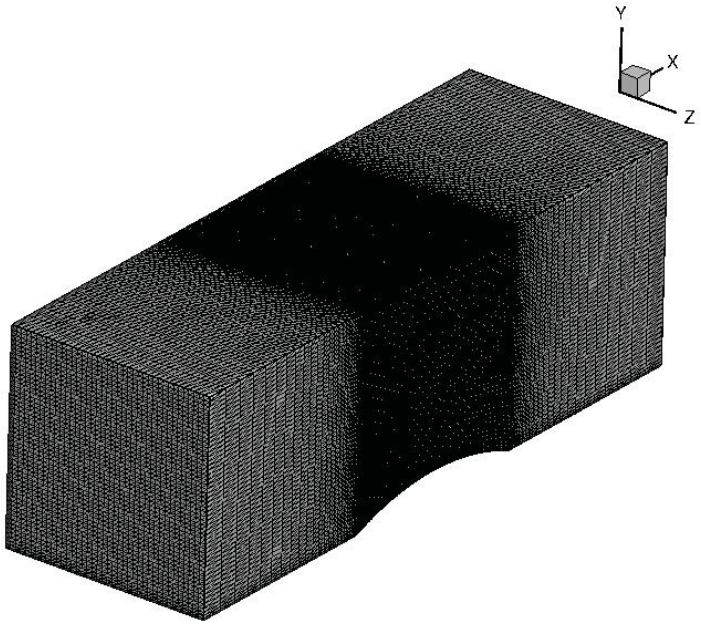


Figure 5.2: Bump' mesh.

converge with more or less the same slope. However, the slope is sharper in the case that we increase the outer loop. The simulation with (nbnewton=8) converge after 200 time steps (iterations), while the case with (nbnewoton=1) converges only after 600 time steps. This is expected since the outer loop correct the deviation of the approximated Jacobian Matrix at each iteration. However, more outer loops means increased CPU times. Table 5.2 shows the number of inner iterations and CPU time inside the linear solver for each outer iteration for (nbnewton=8) in order to reach the linear convergence criteria. It is seen that the total CPU time from the third to the last iteration combined is less than the summation of first two iteration. It also can be observed that CPU time inside the linear solver is linearly proportional to the number of iteration of the linear solver.

Figure 5.6 compares the L2-norm residual of four simulations with different nbnewton values. The discrepancy between one Newtons' iteration and two is significant. There is a small difference between (nbnewton=1) and (nbnewton=2) and it is almost negligible between (nbnewton=4) and (nbnewton=8).

In Figure 5.6 and Figure 5.7, it can be seen that the residual at the beginning of the simulation increases, which is related to the sharp slope of CFL number. The CFL number at each time step is calculated from following equation,

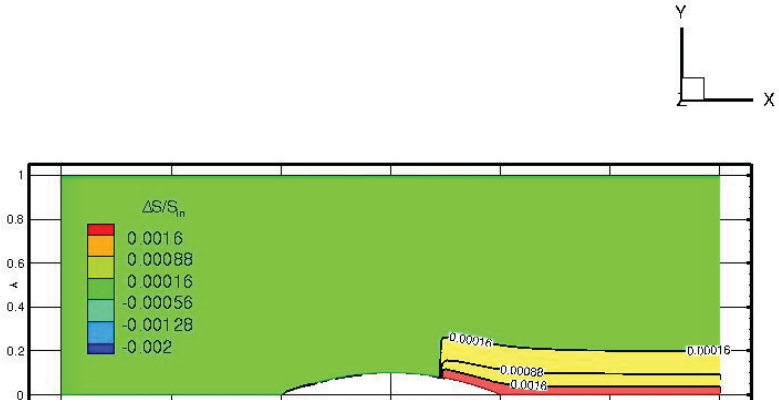


Figure 5.3: Entropy's contour over the transonic circular arc bump.

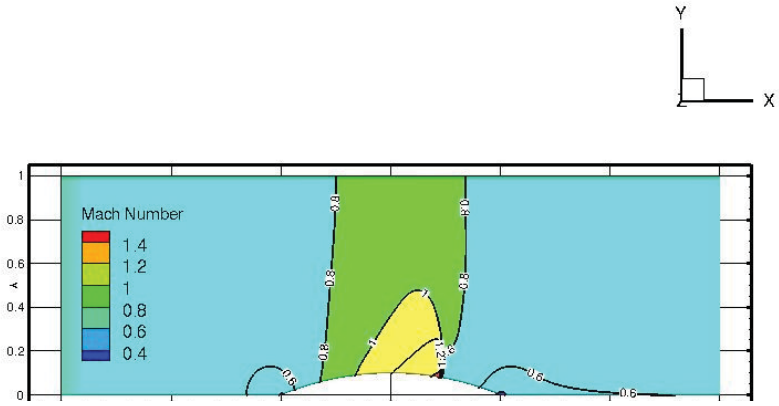


Figure 5.4: Mach number contour over the transonic circular arc bump.

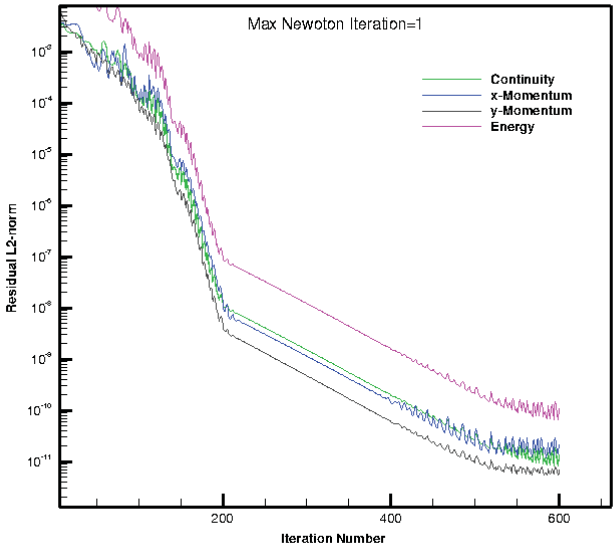
$$CFL^{n+1} = CFL^{n}.\frac{||\Delta \mathbf{W}^{n-1}||_2}{||\Delta \mathbf{W}^{n}||_2} \tag{5.1}$$

where $||\Delta \mathbf{W}^{n-1}||_2$ and $||\Delta \mathbf{W}^{n}||_2$ are the second norm of the residual in the previous time steps. In Figure 5.8, the CFL number starts from one in all the simulations and it increases above 800 for (nbnewton=2) after 500 time steps. It shows that our implicit solver is robust even for the large CFL numbers on a uniform grid.

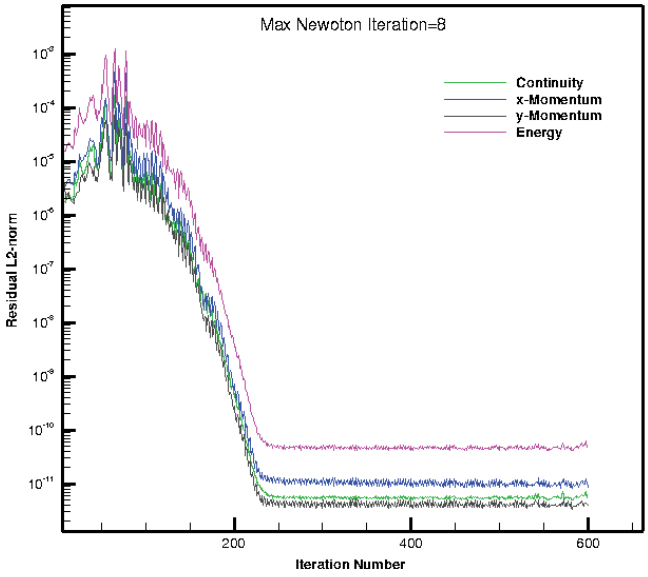
Last to investigate in this section is the outcome of changing the restart parameter in GERMS is investigated (Algorithm 1 in chapter 3). In this study, nbnwton=2 and CFL number are increased according to equation (5.1), where it is noticed that its impact on convergence is negligible as seen is Figure . Therefore for larger problems we may use a restart of 10 to reduce the memory usage.

Table 5.2: Compare CPU time and linear solver iteration at different outer iteration for (nbnewton=8).

Outer Loop Number	CPU (Seconds)	Linear Solver Iteration
1	3.56	151
2	1.86	80
3	1.35	48
4	0.82	31
5	0.66	24
6	0.52	20
7	0.38	16
8	0.33	12



(a)



(b)

Figure 5.5: The L2-Norm residual; (a) nbnewton=1; (b) nbnewton=8.

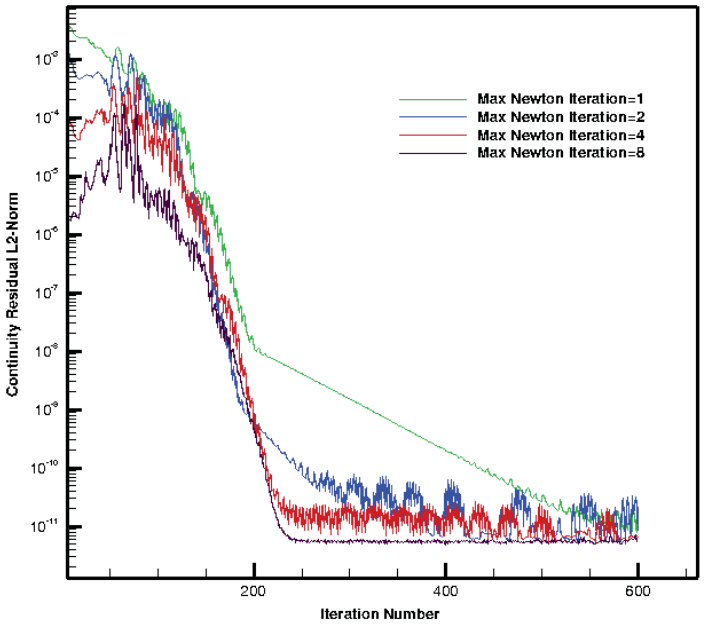


Figure 5.6: Continuity residual for a transonic bump.

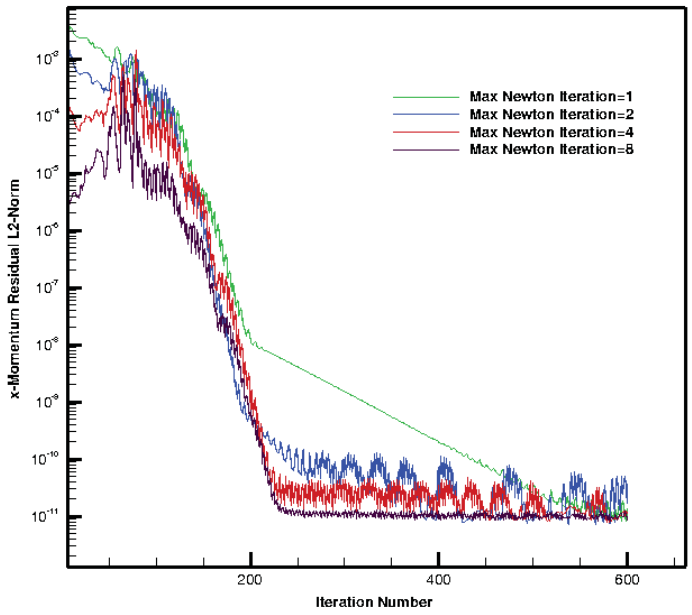


Figure 5.7: The x-Momentum residual for a transonic bump.

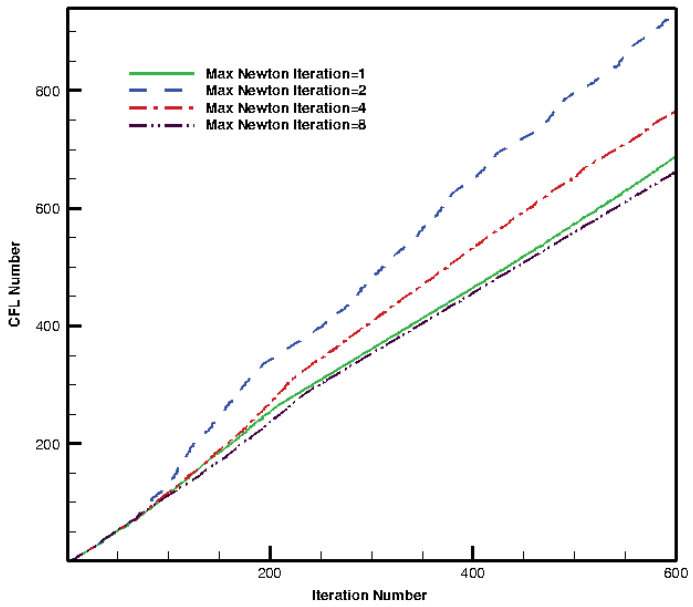


Figure 5.8: Trend of the CFL number versus iteration number for a transonic bump.

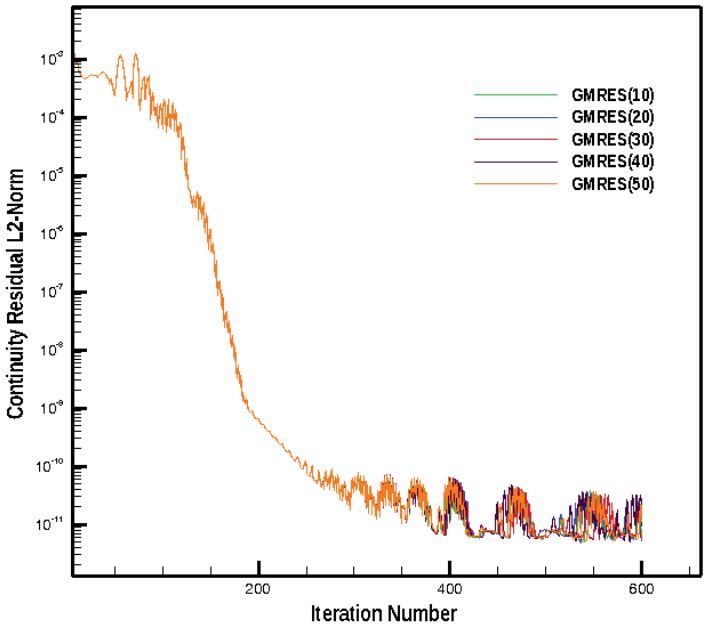


Figure 5.9: Impact of restart parameter on GMRES convergence.

5.2 Subsonic Circular Arc Bump

In this section, convergence study is extended to compressible subsonic flows. The Bump's geometry and boundary conditions are similar to the transonic case, except the velocity inlet condition is changed in order to have a subsonic flow in the entire domain (Table 5.3). The numerical schemes are also the same as in the transonic bump case, except for the stopping criterion of the linear solver, which is set to $O(10^3)$. First, the effect of the cell quality on the convergence is studied. Next, the impact of CFL number on the slope of the convergence is investigated.

Table 5.3: Boundary conditions and geometry properties for subsonic flow over bump.

Parameters	**Mach Number**	**Bump Size**
Value	0.5	10 %

Figures 5.10 and 5.11 show the contours of Mach number and pressure distribution in the domain. The C_p in Figure 5.11 is the pressure coefficient. There is a good agreement between the results presented here to those shown in references [27, 32].

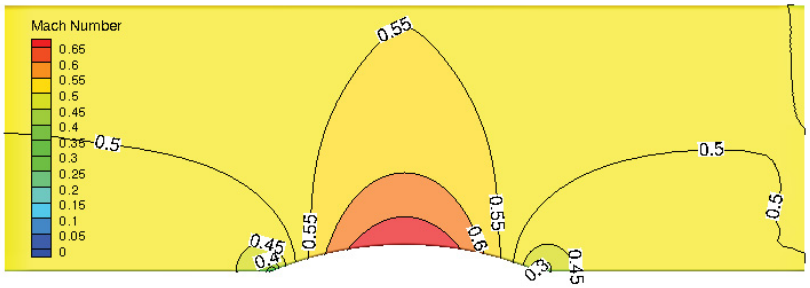


Figure 5.10: The Mach distribution over the subsonic bump.

Figure 5.12 shows the convergence rate of the L2-norm of the residual at different number of non-linear iterations (nbnewton) for a uniform cell. The convergence is improved by increasing the number of the Newtonian iterations. An acceptable slope even at the (nbnewton=1) is observed. This is due to the fact that subsonic flows does not experience any discontinuity such as the shock in the transonic case, and in general, a better convergence can be observed for subsonic flow.

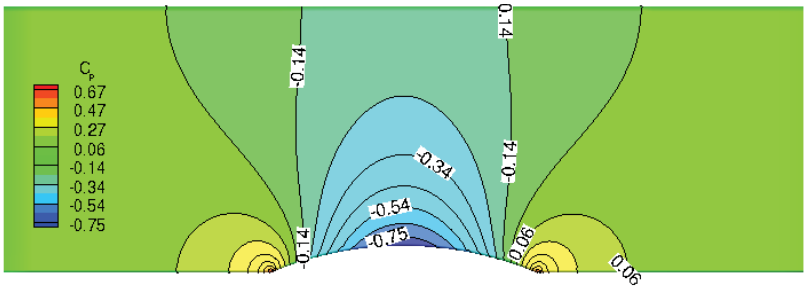


Figure 5.11: The pressure distribution over subsonic bump.

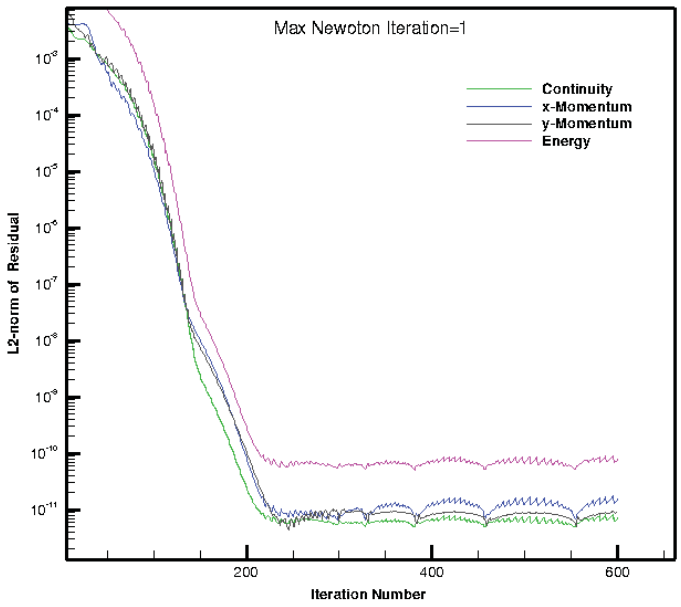
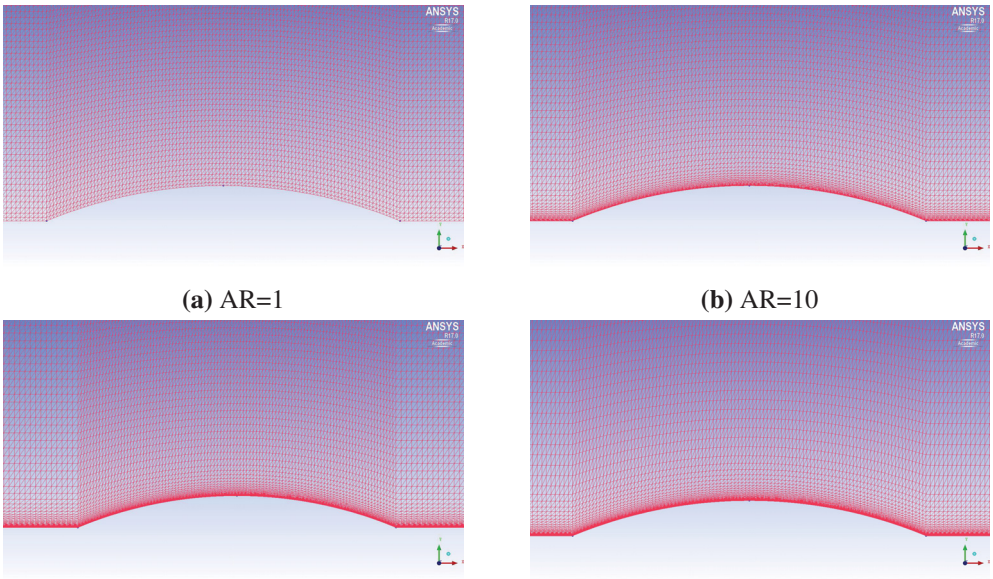


Figure 5.12: The L2-Norm residual for a subsonic bump (nbnewton=1).

Four levels of cell aspect ratio on the bump surface are investigated to compare the convergence sensitivity to the grid quality. Four grid meshes are built (Figure 5.13) with cell aspect ratio of 1, 10, 30, and 100. The aspect ratio is defined as ratio of length of the largest edge to the smallest one in a cell. In order to generate those meshes, hexahedron meshes are first built where their individual cells are then are split into six tetrahedron. Although, creating a boundary layer mesh for an Euler problem may seem unnecessary, this strategy is chosen since the final goal is to simulate viscous

simulations. The boundary layer simulations will require very fine meshes close to the surface which results in stretched elements. Therefore, for the sake of simplicity, the aspect ratio definition is based on the hexahedron element dimensions. The aspect ratio are applied both in the normal and the span-wise direction as it is shown in Figure 5.14.



(a) AR=1 **(b)** AR=10

(c) AR=30 **(d)** AR=100

Figure 5.13: Front view of mesh on the subsonic bump.

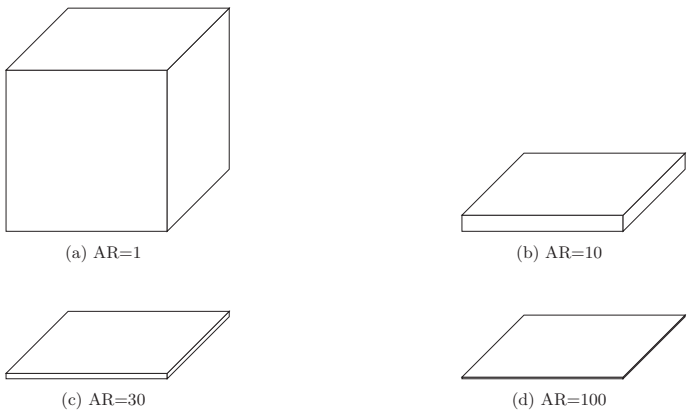


Figure 5.14: Hexa cell aspect ratio.

Figure 5.15 compares the convergence of the solver with meshes of different aspect ratios. As

it is expected, the uniform grid gives the best convergence rate. On the other hand, the solution diverges if only one newton iteration at (AR=30) and (AR=100) is used. One may prevent the divergence due to the grid for (AR=30) by increasing the non-linear solver iterations. However, the residual only drops two orders of magnitude (Figure 5.16). In the case of AR=100, no matter how many Newton's iterations are used, it always crashes as long as the CFL number grows.

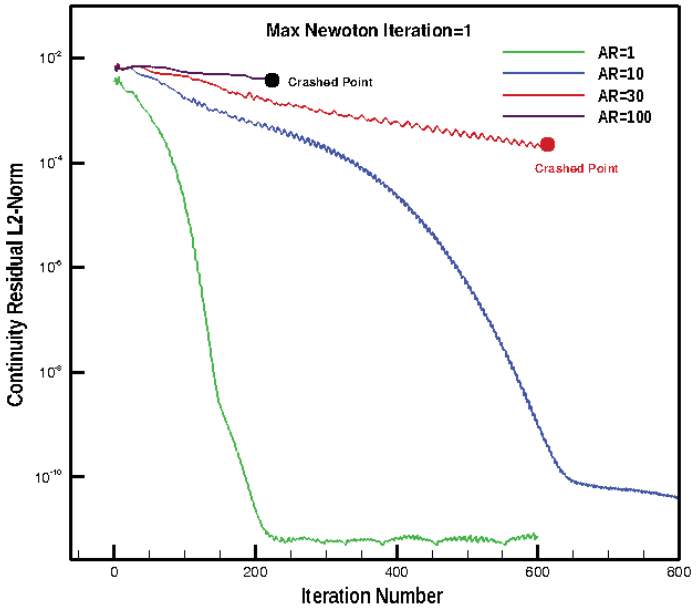


Figure 5.15: Comparison cell quality on convergence (nbnewton=1).

In the previous simulations, no cap was set for the CFL number which increases according to equation (5.1). In Figure 5.17 a cap for the maximum CFL number is set for the case of AR=100. The results show that adding a cap can prevent divergence, however, the solutions still lacks a good convergence value. Smaller CFL numbers have another disadvantage, where it needs more time step to reach to the steady state solution, and it may increase the round off error as well.

A structured mesh is less sensitive to the aspect ratio in general. On the other hand, a convergence problem appears for unstructured meshes with boundary layer grids in the near-body region. The skewness of an element, similar to a cell shown in Figure 5.18, can be given by,

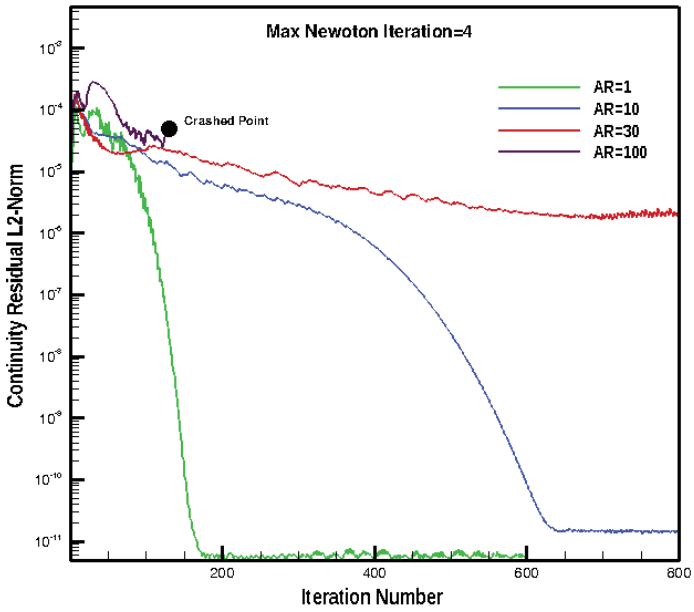


Figure 5.16: Comparison cell quality on convergence (nbnewton=4).

$$Skewness = Max(\frac{\theta_{max} - 90}{90}, \frac{90 - \theta_{min}}{90}) \tag{5.2}$$

where θ_{max} and θ_{min} are largest and smallest angles between the edges of a cell. Many commercial CFD codes converge when the maximum skewness in the domain does not exceed 0.9, and majority of the cells' skewness fall below 0.6. Consider a quad or hexahedral cell that is completely orthogonal, similar to cells shown in Figure 5.14. The skewness of a triangle that is built by splitting this quad is summarized in Table 5.4. The skewness is very high for all the aspect ratio except AR=1. However, the current in-house code is robust and converge with skewness less that 0.96, which is superior to many commercial codes such as FLUENT. Still, it is better to avoid increasing the aspect ratio of boundary cells more than 20 for the Navier-Stokes simulations.

5.2.1 Comparison of Matrix-free with Jacobian-based solver

In this section the convergence of the subsonic bump is investigated by using two strategies. All the results so far have calculated the Matrix Jacobian of the fluxes. One may use a simpler approach

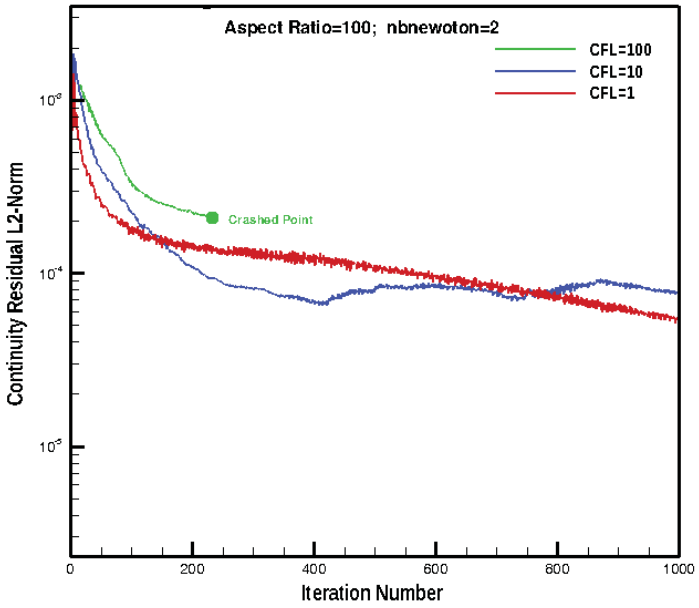


Figure 5.17: Effect of CFL number on the convergence (AR=100).

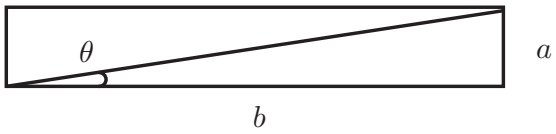


Figure 5.18: Skewness angle.

by using a Matrix-free solver. Then, only the diagonal part of the Jacobian is constructed in order to be used as a preconditioner. Figure 5.19 compares the convergence of these approaches. For both methods, the CFL number grows to reach around 1000. Results are studied for four levels of the Newtons' iterations. It is seen that a solver that build the Matrix-Jacobian of the fluxes results into a better convergence. Matrix-free diverges when only one non-linear iteration is used. Since the slope is approximated with the linear Taylor expansion in the Matrix-free solver, the slope of fluxes calculated by this method is deviated from the actual derivative of the fluxes' terms. This situation worsens when the CFL number increases. However, discrepancy between two approaches almost vanish as the Newtons' iterations increase.

Figure 5.20 compares the CFL number and the required inner loop iterations to converge to the criteria of the linear solver. The results are shown for both the Jacobian-based and the Matrix-free

Table 5.4: Comparison of cell skewness of a triangle

Aspect Ratio	θ_{min} (Degrees)	Skewness
1	45	0.5
10	5.7	0.93
30	1.9	0.98
100	0.57	0.993

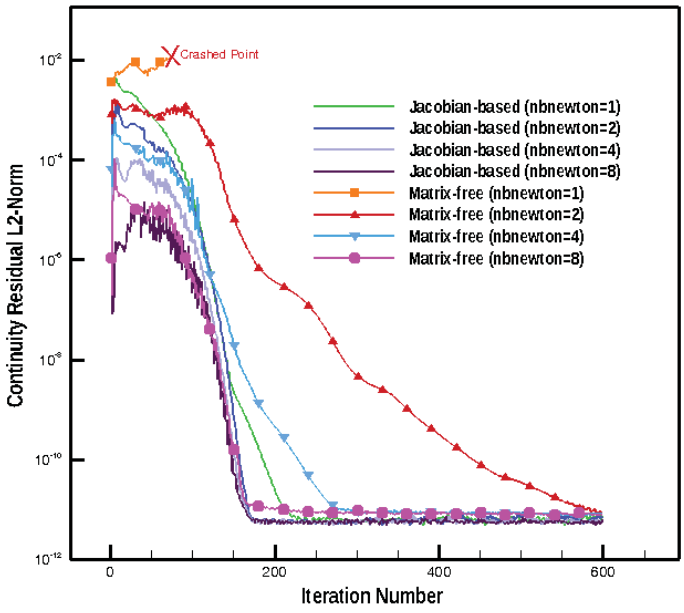


Figure 5.19: Comparison of the convergence between the Jacobian-based and the Matrix-free.

methods. This shows that, due to the larger deviation between the slope and the actual fluxes' derivative in the Matrix-free, more inner iteration is required inside the linear solver. This is true even if the CFL number is higher in the Jacobian-based model. After 210 time step, solution converges to the steady states solution,and the number of inner loop significantly drops. Finally, the required CPU time for the same number of inner loop for the Matrix-free solver is almost double of the Jacobian-based model. This is due to the fact that in the Matrix-free solver, the derivative needs to be updated at each inner loop, while it is only calculate once at each time step in the Jacobian-based.

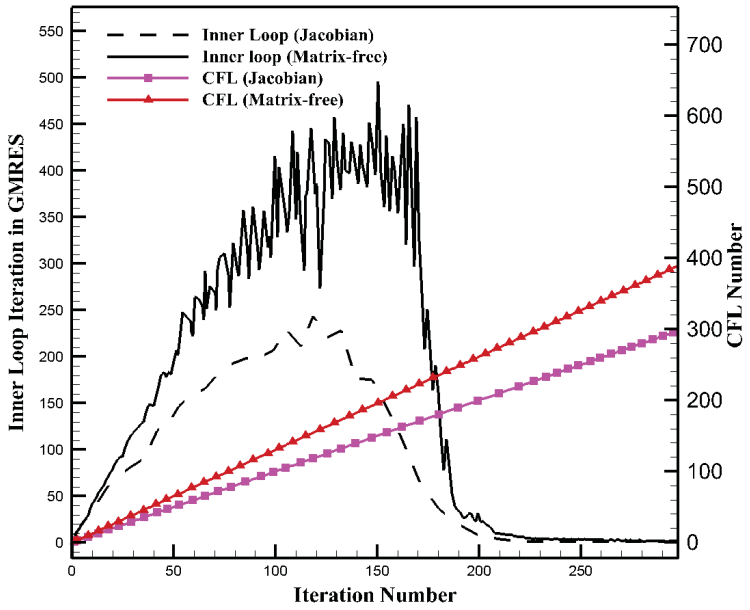


Figure 5.20: Comparison of number of inner loop and CFL number between the Jacobian-based and the Matrix-free.

5.3 Analysis of the efficiency of GMRES inside the In-house Code

The Portable, Extensible Toolkit for Scientific Computation (PETSs) is an open-source package that solves linear systems [5] for large sparse matrices as well as medium dense ones. The package uses different preconditioning methods such as Algebraic Multi-Grid (AMG), ILU, Jacobi, and etc. The current in-house code has been integrated with the PETSc through its FORTRAN interface (The interface subroutine is shown in Appendix B). The goal is to compare the robustness of our in-house code solver and PETSc's, as well as provide access to a broad range of available preconditioning methods and solvers in the PETSc.

Table 5.5: Inlet condition and geometry property for transonic flow over NACA0012

Parameters	Mach Number	Chord Length
Value	0.8	0.5

The geometry and the boundary conditions of NACA 0012 is shown in Figure 5.21. The domain

size is $1.5 \times 1.5 \times 0.1$ and the chord and the inlet conditions are given in Table 5.5

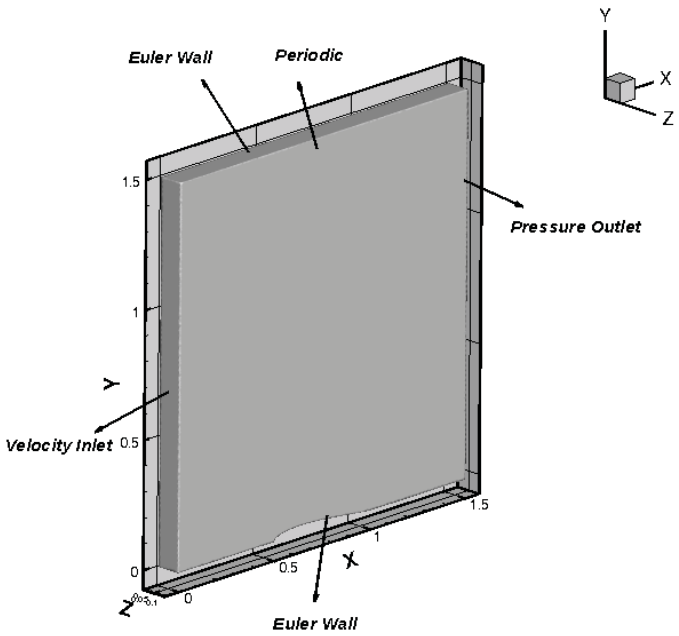


Figure 5.21: The geometry and boundary conditions of transonic Euler NACA 0012.

The Mach number contours are displayed in Figure 5.22. As seen, the Mach number increases in the first half and keeps accelerating in the second half until the shock appears, where the flow then returns to the subsonic regime.

Figure 5.23 compares the convergence between using the GMRES inside the PETSc and the in-house code version of GMRES. It is seen that both results are very similar. That proves the robustness of the implementation of GMRES in the in-house code. It also means that the Matrix of coefficients are constructed correctly within the PETSc.

Regarding the CPU time, PETSc may need more computational time since mapping of the index numbering from the natural numbering to the PETSc's format is needed as illustrated in Figure 5.24. Indexing in the application code is usually dictated by the partitioning method for example METIS is used in the in-house code. The global numbering of the nodes does not follow a certain pattern among the partitions, while in PETSc, global number starts from one in the first partition and ends to the last node in the final partition in order to reduce the communication between the partitions. Solving a linear system in PETSc includes three main phases: Map and store the matrix coefficient

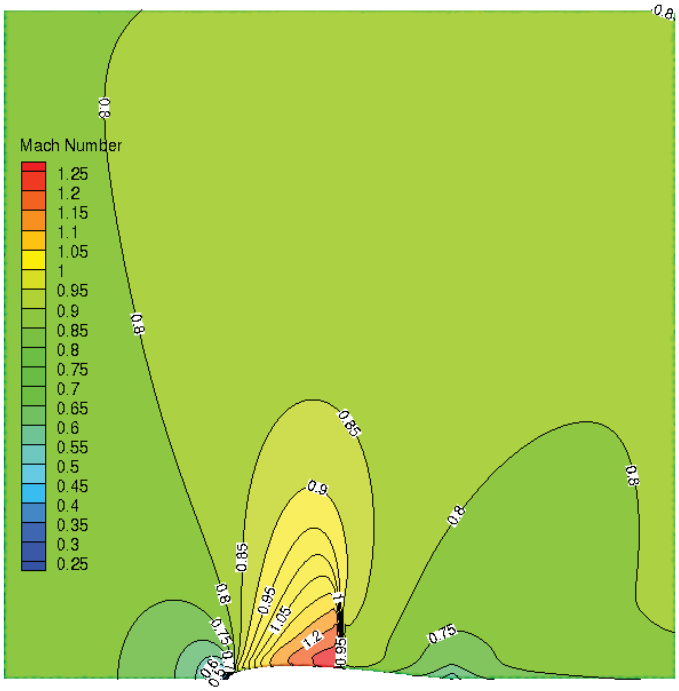


Figure 5.22: The Mach contours over transonic Euler NACA 0012.

and vectors in PETSc's format; Solve the system with PETSc; Return the linear-solver solution to the in-house code in order to continue the simulation in the next Newtons'loop or time step. Our study shows that for a large linear system, time required to build the matrix may even exceed the CPU time for solve the system by one order of magnitude. Therefore, we use the in-house code version of GMRES solver for the Navier-Stokes equations. According to results we collected, we perform the rest of our simulations with our in-house code GMRES that solve the system in a shorter time.

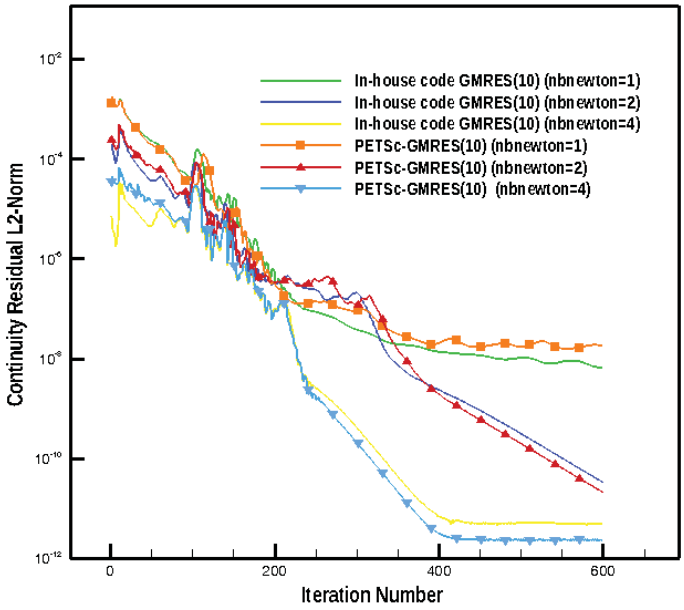


Figure 5.23: Comparison of convergence between PETSc and in-house code solver.

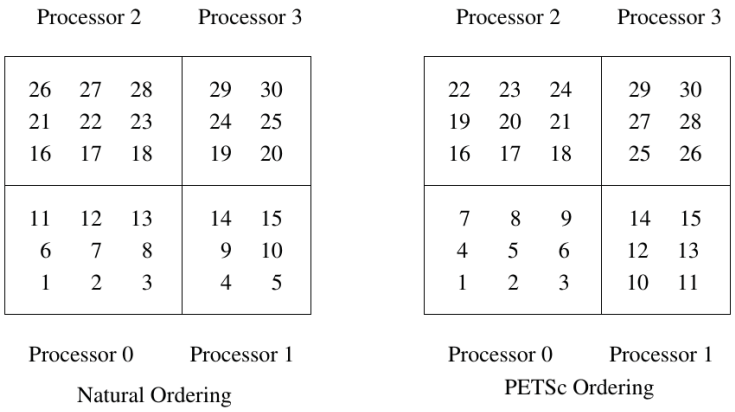


Figure 5.24: Natural and PETSc numbering [5].

5.4 Laminar Flow in Pipe

In order to investigate the convergence of the Navier-Stokes equations, a laminar pipe is considered. The second order Roe-MUSCL with the Van-Albada limiter for the spatial discretization as well as an implicit scheme with the second order time accuracy is used. The GMRES solver is used

to solve the linear system with a relative residual limit inside the linear solver of $(O(10^{-3}))$.

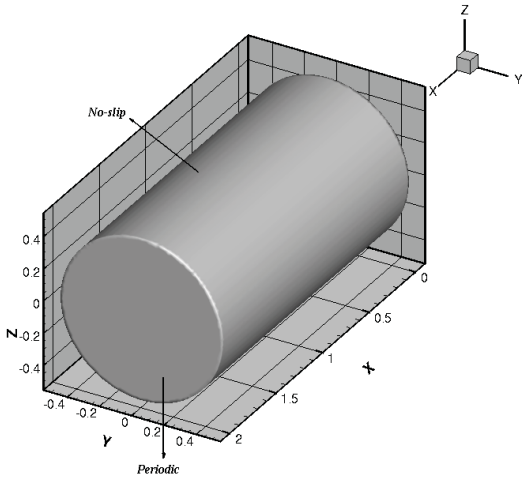


Figure 5.25: Pipe's geometry and boundary condition.

In order to mimic the pipe flow, periodic boundary conditions are applied in the stream-wise direction and a source term is added to model the pressure drop. Flow is considered laminar for $(Re < 2300)$ inside a pipe. The pressure drop is given by,

$$\frac{\Delta P}{L} = \frac{64}{Re}\frac{\rho V_{ave}^2}{2} \tag{5.3}$$

Table 5.6: Inlet condition and geometry property for laminar pipe.

Parameters	**Mach Number**	**Reynolds Number**
Value	0.25	1000

where Re is calculated based on the bulk velocity. Flow properties are given in Table 5.6. The pipe length is twice of the pipe's diameter as seen in Figure 5.25. The grid includes 468909 points and 2733782 tetrahedral elements as shown in Figure 5.26. As seen in Figure 5.27 the flow is completely developed in the pipe and no variation is observed at different cross-sections. There is a very good match between the numerical and analytical solution,

$$u(r) = 2V_{ave}(1 - \frac{r^2}{R^2}) \tag{5.4}$$

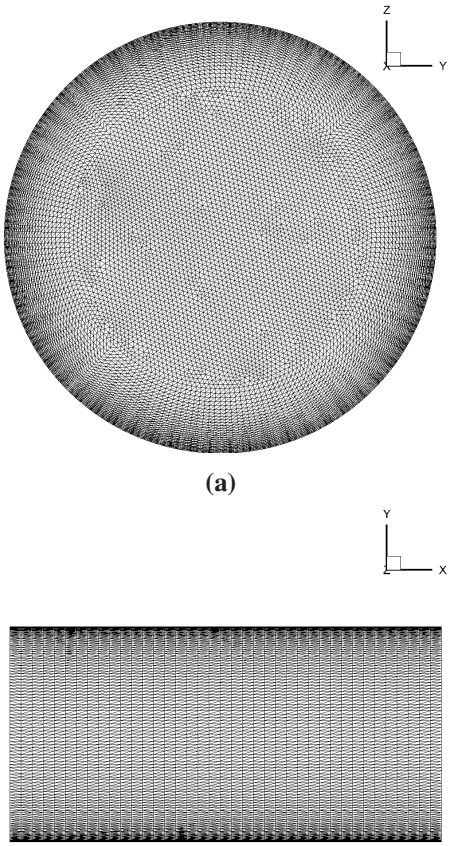


Figure 5.26: Unstructured grid for the laminar pipe flow.

which also predicts the maximum velocity $V_{max} = 2V_{ave}$. Figure 5.29 repeats the convergence history for four different outer loop iteration values. The difference between the number of outer loops is very drastic in comparison to the Euler bump case. Therefore, at least two Newtons's iterations are required to improve the convergence.

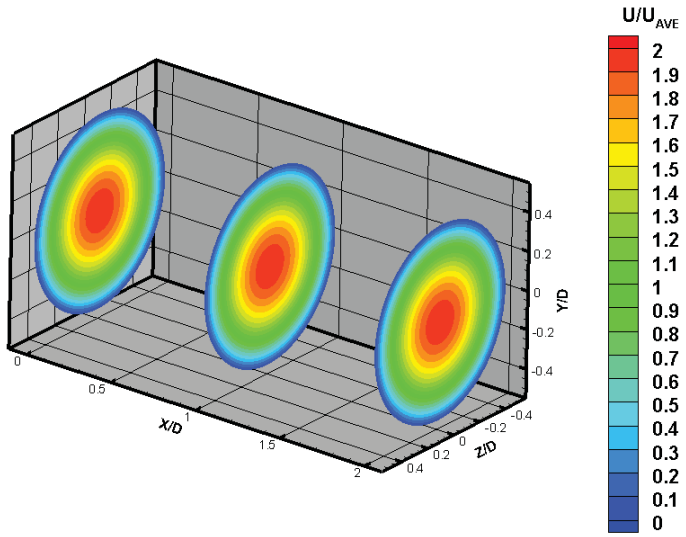


Figure 5.27: Velocity contour at the ends and the middle plain of the pipe.

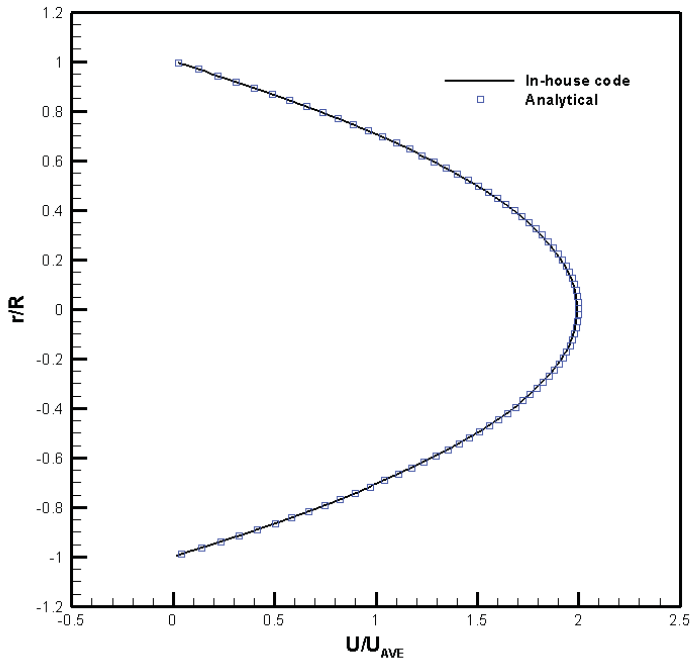


Figure 5.28: Comparison of the analytical and numerical solutions for the laminar pipe flow.

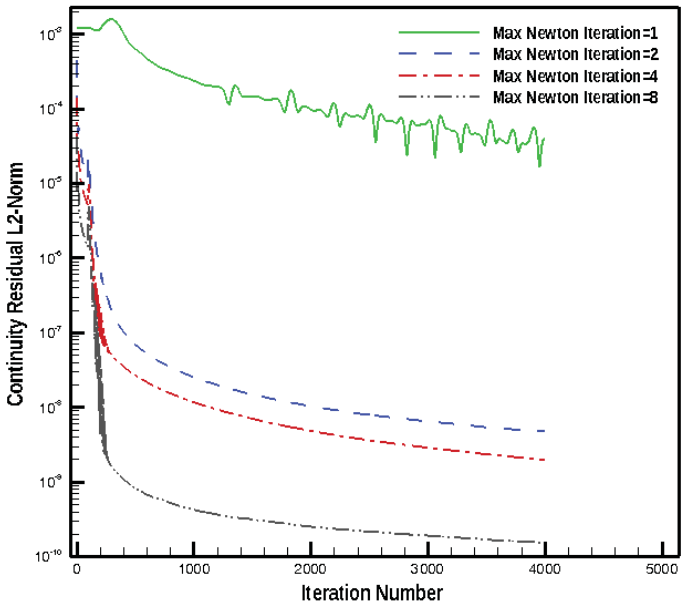


Figure 5.29: The convergence of the laminar flow in the pipe.

5.5 Deformation Study

In this section, the robustness of the deformation methods that were described in the previous chapter is investigated. Herein, focus is placed only on the deformation method, and not the flow solutions. For this purpose, two test cases are selected: A cylinder that goes under expansion and contraction, and a deflecting Cantilever beam.

5.5.1 Cylinder undergoing expansion and contraction

A cylinder with an initial radius of $(R_i = 0.5)$ and height of $(H = 0.1)$ is chosen. The far-field boundary is set to $10 \times R$ form the center. The radius oscillate between $(1.5 \times R)$ to $(0.5 \times R)$, while its height remains unchanged. A forced displacement is applied on the points on the cylinder,

$$R = R_i + \frac{R_i}{2} \sin(\frac{2\pi n}{N}) \tag{5.5}$$

where N is the period of oscillation and n presents the time step. The interior points are adjusted

to the their new location with the spring and diffusion schemes. The results are presented for two types of grid: Uniform and stretched grid .

Figure 5.30 compares the quality of mesh when it is updated by the spring and the diffusion methods for the uniform case. Both methods calculate the new location of grids with the minimum variation to the orthogonality and the volume of cells. The results are not surprising as the spring method is based on the linear deformation of a spring, and the grid displacement of both expansion and contraction only appears linearly along most of the edges. The diffusion method is also very robust, while a linear deformation is expected.

The stretched grid is used with AR=10 for the next test case. It is also shown that both schemes preserve the quality of mesh (Figure 5.31). The volume changes to 1.44 and 0.6 times of its initial value for expansion and contraction, respectively, which means that the cells' length in the normal and span-wise directions almost remained unchanged. The solution is repeated with different values of $(\alpha \in \{1.0, 1.5, 2.0\})$ and $(C_{ex} \in \{0.1, 0.5, 1.0\})$ defined in equations (4.8) and (4.7) in chapter 4, and no significant change has been observed.

5.5.2 Cantilever beam deflection

A cantilever beam is a beam that is fixed at one end, and has a concentrated load applied to the free end (Figure 5.32). It is one of the classic problem in the structural analysis, and the deflection at each point of the beam is given by,

$$\delta(x) = \frac{F}{6EI}(-x^3 + 3Lx) \tag{5.6}$$

and it depends on the concentrated force, F, the moment of inertia of cross surface (here an square with edge size b), $I = b^4/12$, Young's modulus, E, and length of the beam, L. In order to preform the deformation, the force is oscillated between -P to P with sinusoidal function,

$$F = P\sin(\frac{2\pi n}{N}) \tag{5.7}$$

where P is the maximum concentrated load. The geometric and physical properties of the beam are tabulated in Table 5.7.

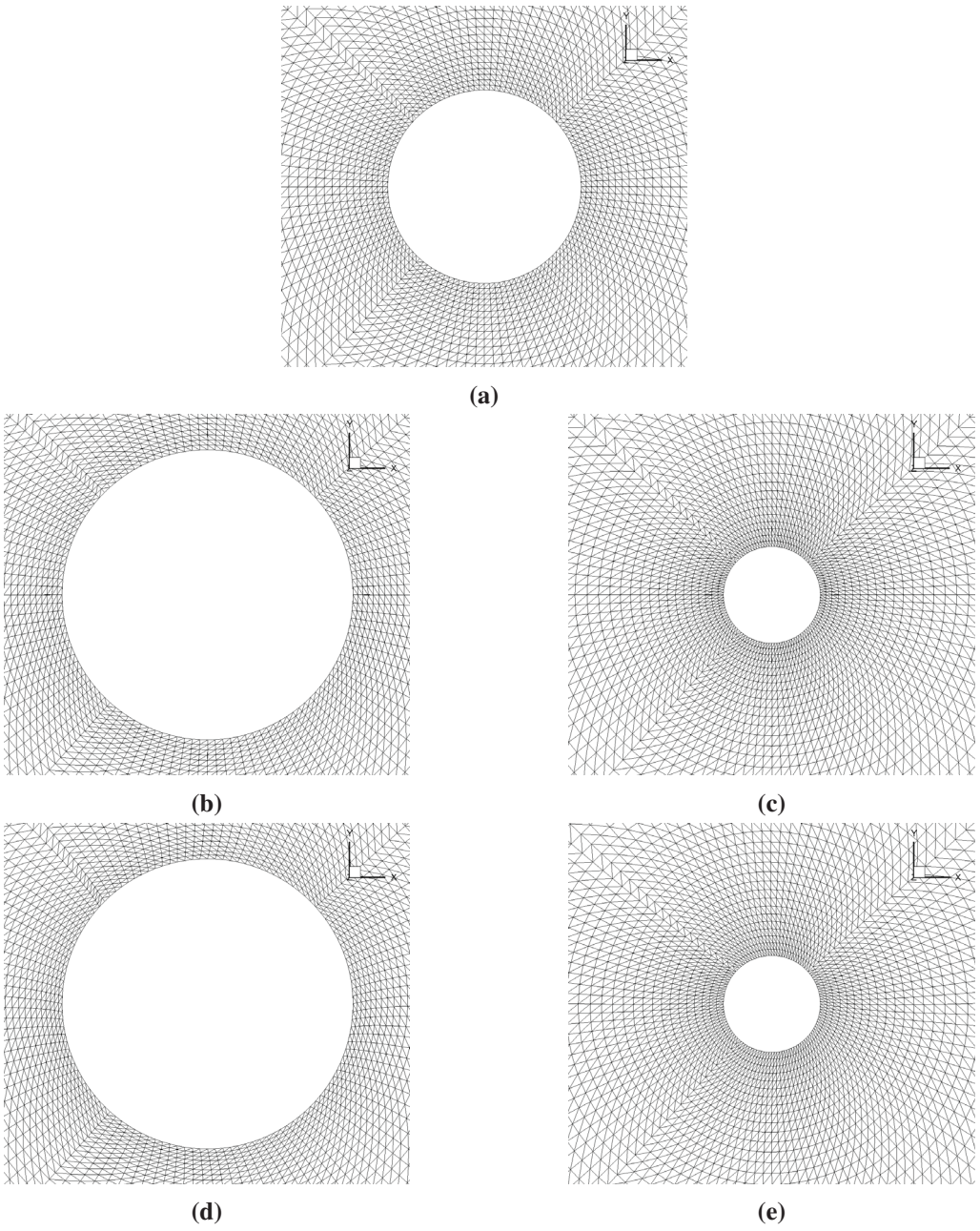


Figure 5.30: (a)Initial grid; (b) Diffusion $\gamma = 1/d^2$, $R = 1.5 \times R_i$; (c) Diffusion $\gamma = 1/d^2$, $R = 0.5 \times R_i$; (d) Spring, $\alpha = 2$, $C_{ex} = 0.5$, $R = 1.5 \times R_i$; (e) Spring, $\alpha = 2$, $C_{ex} = 0.5$, $R = 0.5 \times R_i$.

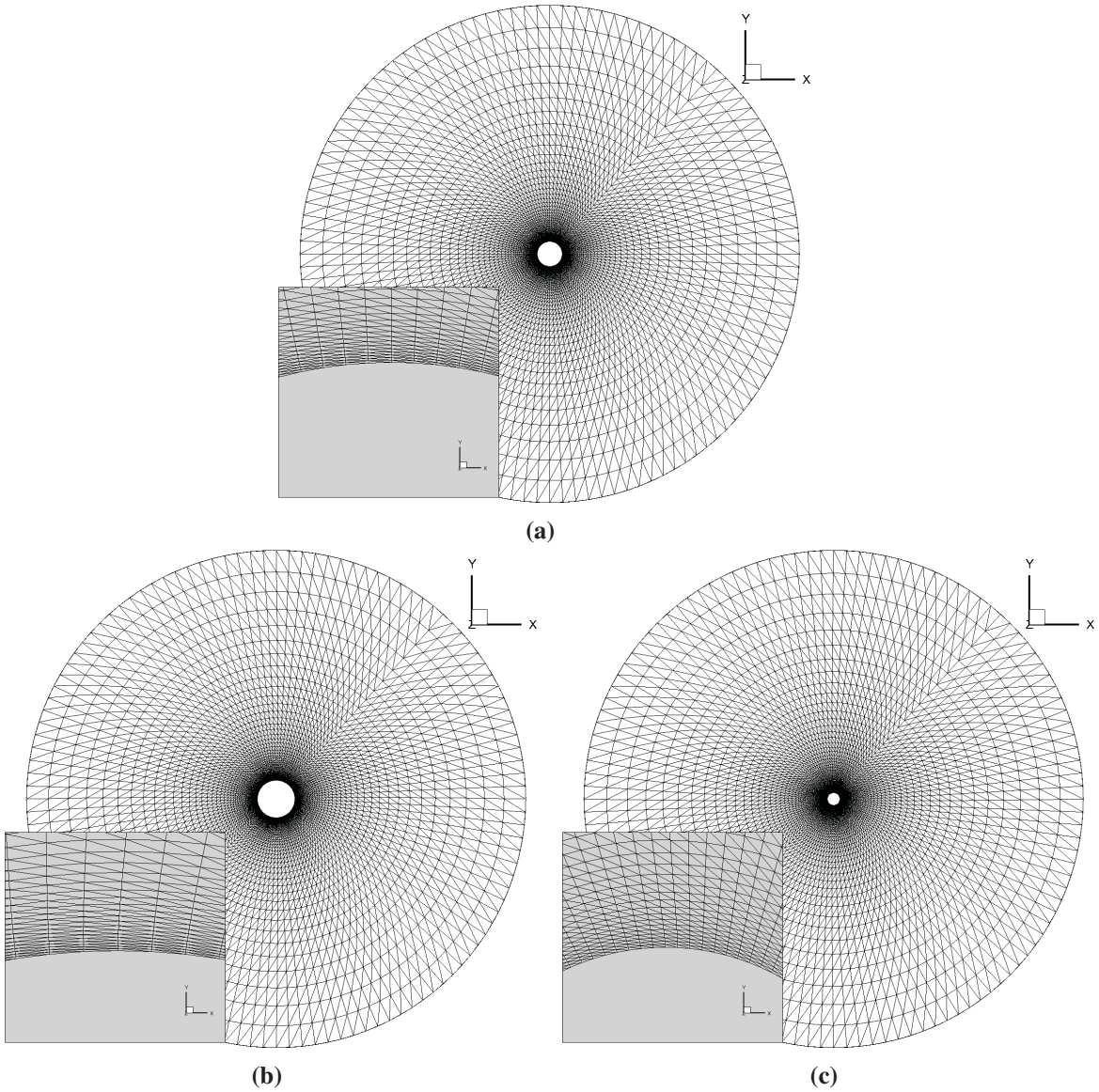


Figure 5.31: Stretched Grid. (a) Initial grid; (b) Diffusion $\gamma = 1/d^2$, $R = 1.5 \times R_i$; (c) Spring, $\alpha = 2$, $C_{ex} = 0.5$, $R = 0.5 \times R_i$.

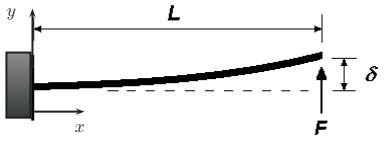


Figure 5.32: Cantilever beam deflection.

Table 5.7: Physical and geometric properties of Cantilever beam.

Length (m)	Cross Section Edge (m)	Young's Modulus (GPa)	Maximum of Load (KN)
3	0.1	69	50

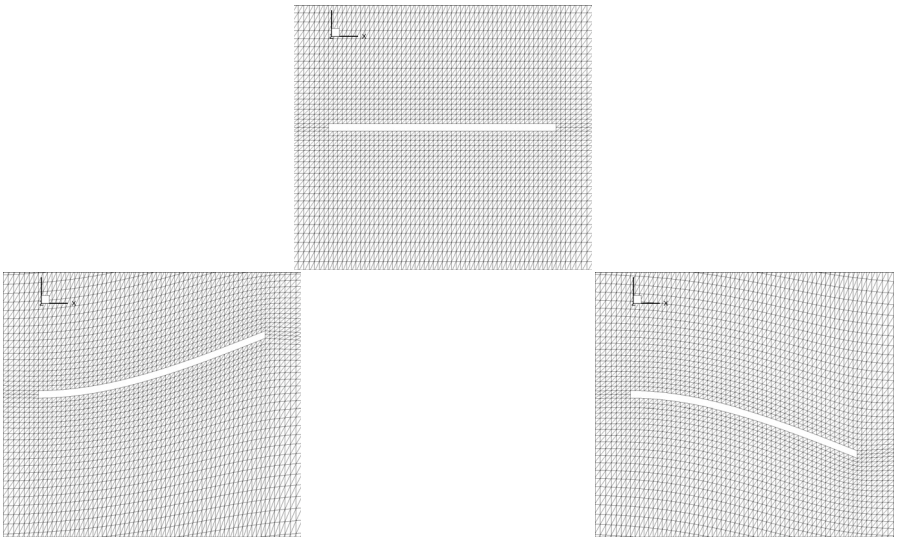


Figure 5.33: Cantilever beam with uniform Grid, Diffusion $\gamma = 1/d^2$, $R = 1.5 \times R_i$.

In this case, nodes do not move along the edges, and they can rotate and transfer at the same time. Therefore, the spring method fails to updated the grid properly. However, the diffusion method is enable to preserve the quality of the mesh both in the case of the uniform grid (Figure 5.33) and the stretched grid (Figure 5.34). The maximum cell volume variation is less than 2% in comparison with its initial value.

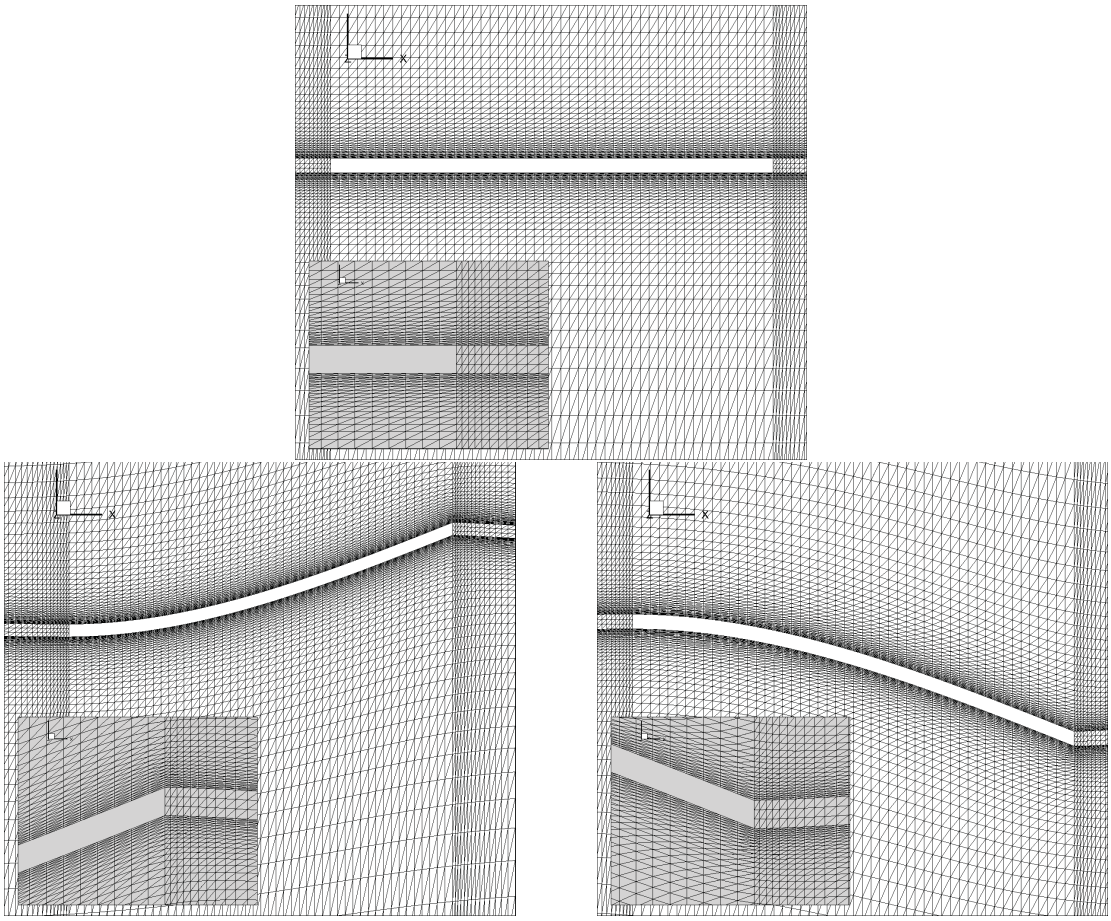


Figure 5.34: Cantilever beam with uniform Grid, Diffusion $\gamma = 1/d^2$, $R = 1.5 \times R_i$.

Chapter 6

Results

First, a validation study is conducted for a flow pass over a standard NACA 0012 airfoil at two high angle of attacks on a 2.5-D mesh. It is critical to ensure that the turbulence methods and the numerical schemes are capable of predicting the physics of the flow, accurately. Afterwards, a Vertical Axis Wind Turbine simulation is conducted with a fixed blade. The focus is on finding the location of dynamic stall as well as the path and patterns of the wake vortices. Finally, capability of the in-house code is investigated to model a morphing airfoil.

6.1 NACA 0012 Airfoil Simulation at High Angle of Attacks

Symmetric blades are very popular for vertical axis wind turbines, mainly because they are easily manufactured and give good performance . Using these type of blade, make the turbine performance independent of the wind direction. This is a key element wherever the wind direction changes frequently. This is common in the urban area or places where tall buildings may influence the wind speed and direction.

6.1.1 Mesh and boundary conditions set-up

The NACA 0012 airfoil is categorized as Four-Digits airfoil profile designed by the National Advisory Committee for Aeronautics (NACA). It is a symmetric blade (dictated by the first two

digits), and the maximum thickness of the blade thickness, represented with the last two digits, is 12% of the chord . The thickness along the chord is given by,

$$\pm\, y(x) = \frac{t}{0.2}(0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4) \tag{6.1}$$

where the maximum thickness is , $t = 0.12$, for NACA 0012 airfoil, and appears at 30% of the chord from the leading edge. Note that the coordinate x changes from zero to the chord size (Figure 6.1). This profile has been used in many applications in a wide range of air planes in the last century. Its main applications in the aeronautics filed includes the US fighter Douglas A2D Skyshark in 1950's, the tip of small size Cessna's wing (Cessna 150, Cessna 187, etc.), and large military aircraft such as Lockheed C-5A Galaxy. In the wind industry, it also has been studied in the variety of VAWT studies [71, 112].

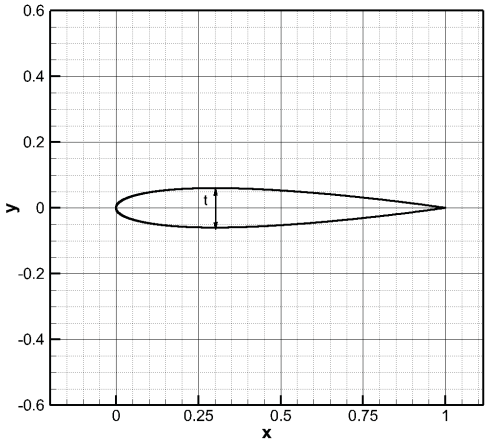


Figure 6.1: NACA 0012 airfoil profile.

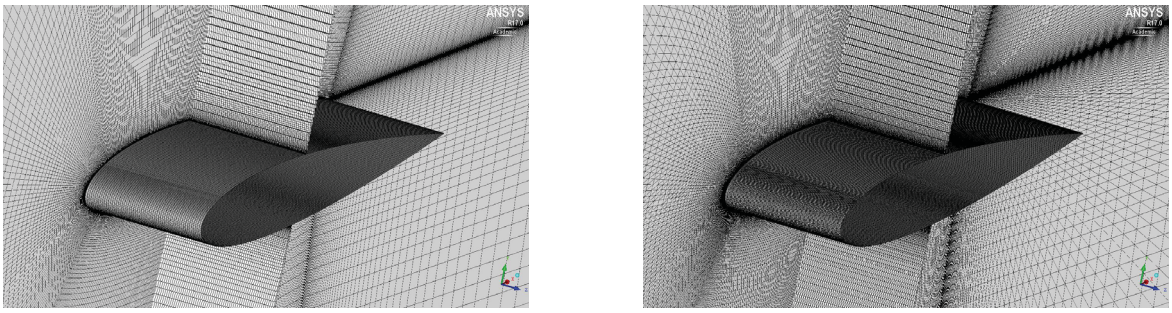
Herein, the flow over a steady blade NACA 0012 at a Reynold number $Re = 50000$ with two angle of attacks $\alpha = 12^\circ$ and $\alpha = 9.25^\circ$ is simulated. Mach number is chosen to be at Mach $= 0.25$. A C-Domain is created with the size of (20$\times$ C) form the blade surface, and it is expanded (0.2$\times$ C) in spanwise direction (Table 6.1).

First, a structured mesh is created in order to have more control on the quality and size of the mesh around the blade surface, then all the hexahedral cells are converted to tetrahedral cells. All

Table 6.1: Flow condition and geometry property of NACA 0012 airfoil

Parameters	Mach Number	Reynolds Number	chord [m]	Span/Chord
Value	0.25	50000	1.0	0.2

of this procedure is done with the software ICEM CFD. The initial hexahedral meshes and the final unstructured grids are shown in Figure 6.2, and detailed information regarding the mesh is summarized in Table 6.2. As seen in the table the aspect ratio near the wall is 10, which based on the previous section gives the best convergence result.



(a) **(b)**

Figure 6.2: (a) Initial structured Mesh; (b) Final unstructured Mesh.

Table 6.2: Mesh information for NACA 0012 airfoil

Cell Type	Tetrahedral
Number of Cells	23890560
Number of Nodes	4107700
$\Delta y_1/C$	0.0003
Geometric Growth Ratio	1.1
x^+/y^+	10
z^+/y^+	10

The periodic boundary conditions is applied on the spanwise faces, and the velocity inlet condition is considered for the left and the bottom surfaces, and pressure far-field for the right and the top surfaces, as it is shown in Figure 6.1.

6.1.2 The numerical set-up

A second order accurate implicit scheme is chosen in time and space. The number of Newtons' loop is set to 2, and the linear residual criteria inside GMRES is set to (10^{-6}). In addition, a Jacobian-based method is chosen to handle the derivative of fluxes. The CFL number eventually increases from 1, and reached to the maximum value of 20. The maximum time step $(\Delta t = 5 \times 10^{-6})$. Selection of these values are also based on the convergence study that is shown in the previous section. The WALE method is used as the turbulence model, and the wiggle detector method is used to control the dissipation of the numerical scheme. The domain is decomposed to the 144 processors using METIS. The computational time to reach $t = 150 U_\infty / C$ is 3 weeks, where C is the chord length. Note that for the sake of comparison a similar domain was tested with OpenFOAM, and it needed to run with one order of magnitude smaller time step in order to preserve the convergence of the solver. Indicating that the in-house code was more and less ten times faster.

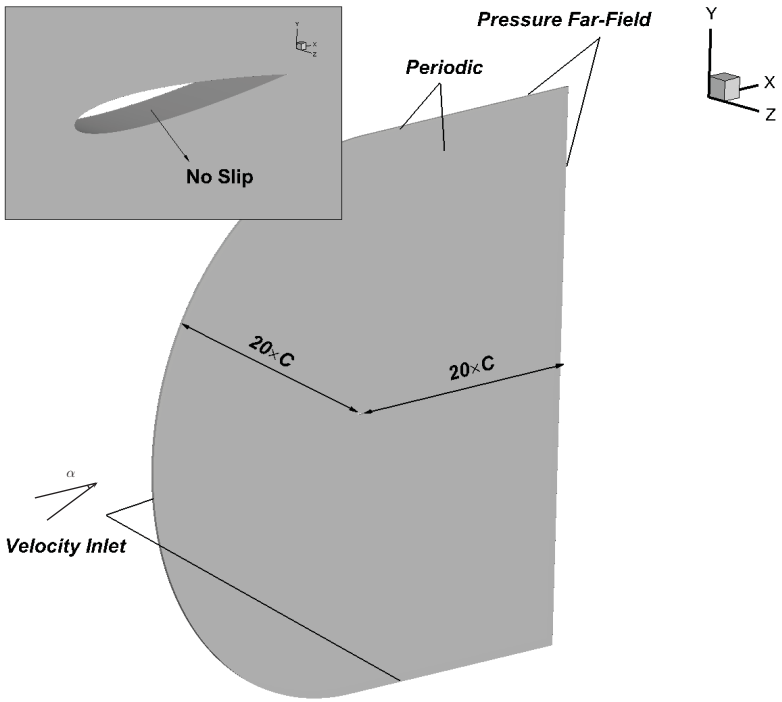
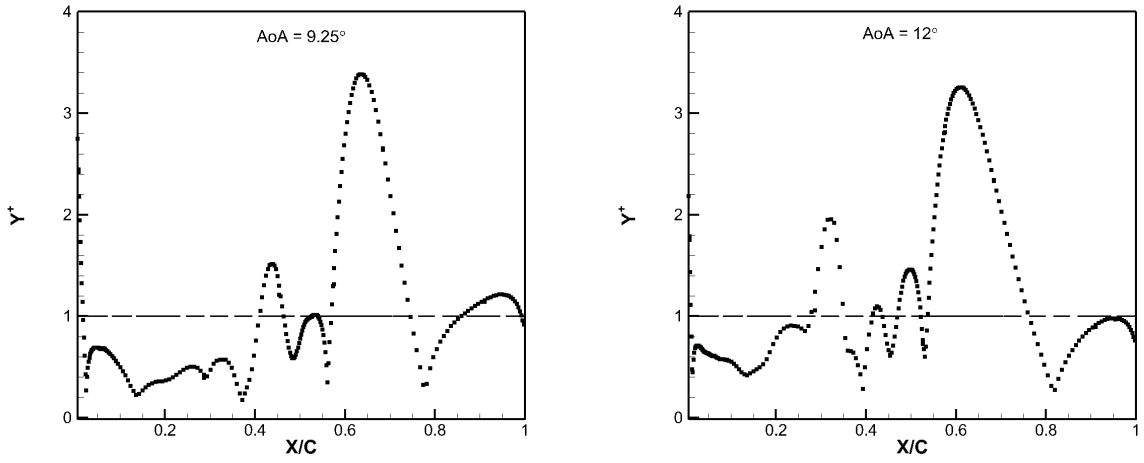


Figure 6.3: NACA 0012 boundary conditions.

6.1.3 Analysis of load on the blade

First, the averaged Y^+ over the suction side of the blade for both angles of attack are calculated to ensure the mesh size inside the boundary layer is sufficient (shown in Figure 6.4). It is seen that over a large portion of the blade, Y^+ falls below 1, and its maximum almost does not exceed 3.

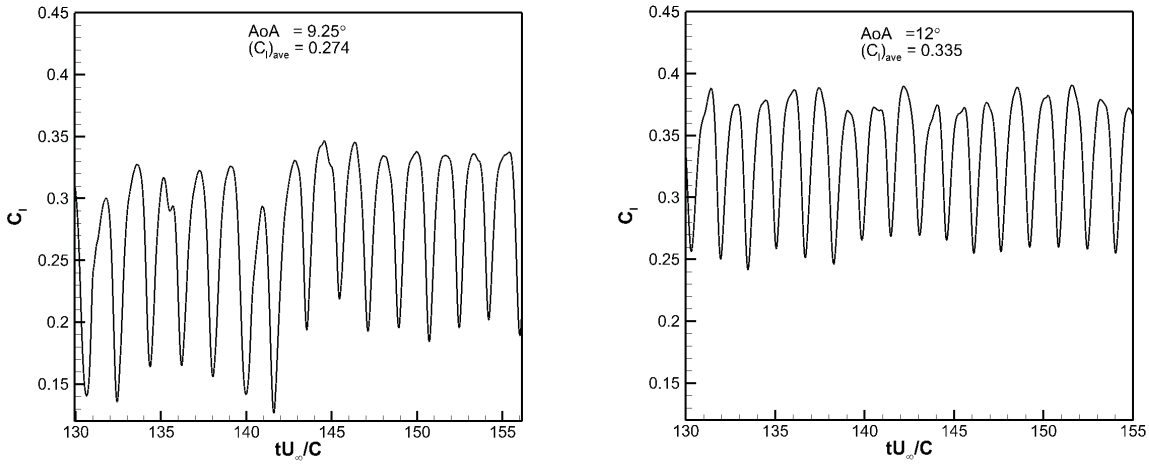


(a) **(b)**

Figure 6.4: Y^+ over the suction side. (a) $AoA = 9.25°$; (b) $AoA = 12°$

This flow at high angles of attack is completely unstable. Due to high shear stress over the suction part of blade, the laminar flow becomes unstable near the leading edge and eventually changes to a completely turbulent downstream. This behaviour is due to the Kelvin-Helmholtz instability. The vortices that are built up on the suction surface, travels to the downstream and finally shed at the trailing edge of the blade. Then a Von-Karman vortex is generated that moves with a certain frequency. The unsteady nature of flow at angles of attack of $AoA = 12°$ and $AoA = 9.25°$ at Reynold number of 50000 is shown in the next section. The visualization the Kelvin-Helmholtz instability and the transition of the flow from laminar to turbulent is elaborated.

Figure 6.5 and Figure 6.6 show the lift and the drag coefficients, respectively. Time is non-dimensionalized with the time of the free-stream velocity. It is observed that the flow can not be considered steady since the load on the surface of the blade changes signification with time. However, the load coefficients follow a pseudo-periodic patterns. The extremums vary from one pick



(a) **(b)**

Figure 6.5: The lift coefficient versus time at: (a) $AoA = 9.25^\circ$; (b) $AoA = 12^\circ$.

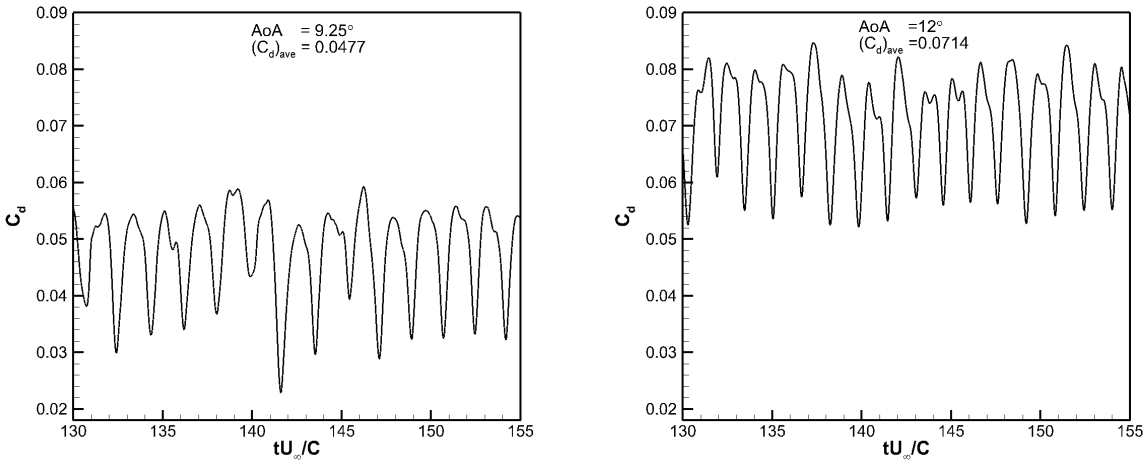
to another. In addition the period of the pattern, which somehow also represents the frequency of the Von-Karman vortex, has a small deviation. The reason is related to the irregular shape and size of vortices that are generated on the blade surface. The average lift coefficient at $AoA = 9.25^\circ$ and $AoA = 12^\circ$ are 0.274 and 0.335, respectively. Then, the lift coefficient increases by 22% from 9.25° to 12° degrees. The average drag coefficient also grows by 50% from the $(C_l)_{ave} = 0.0477$ at the $AoA = 9.25^\circ$ to the $(C_l)_{ave} = 0.0714$ at the $AoA = 12^\circ$. These results is with good agreement with wind tunnel experiment of Sathaye [95] in 2004, where he measured the lift coefficient of 0.26 and o.35 for $AoA = 9.25^\circ$ and $AoA = 12^\circ$, respectively.

Strouhal Number is a non-dimensional parameter that represents the ratio of unsteady inertial forces to the inertial forces due to the velocity variation in space. In other term, it is the ratio of the local acceleration $(\partial u/\partial t)$ to the convective acceleration $(\partial u/\partial x)$. For the unsteady flow over the blade, the Strouhal number is given by,

$$St = \frac{\omega C}{U_\infty} \tag{6.2}$$

where C is the chord length, and U_∞ is the the magnitude of the free-stream velocity, flow oscillate with the frequency , ω. Therefore, the Strouhal Number is the inverse of the aforementioned non-dimensional time (tU_∞/C). Its average at k $AoA = 9.25°$ is $St \approx 0.6$, and it reduces to $St \approx 0.67$ at $AoA = 12°$.

Figure 6.7 shows approximately one cycle of the lift coefficient for each angle of attack, separately. In order to conduct a thorough analysis of the flow, there are six stations that are chosen on the each graph. Contour of pressure, velocity, and vortices, and some 3D-coherent turbulent structures are shown at the marked points in the next section.



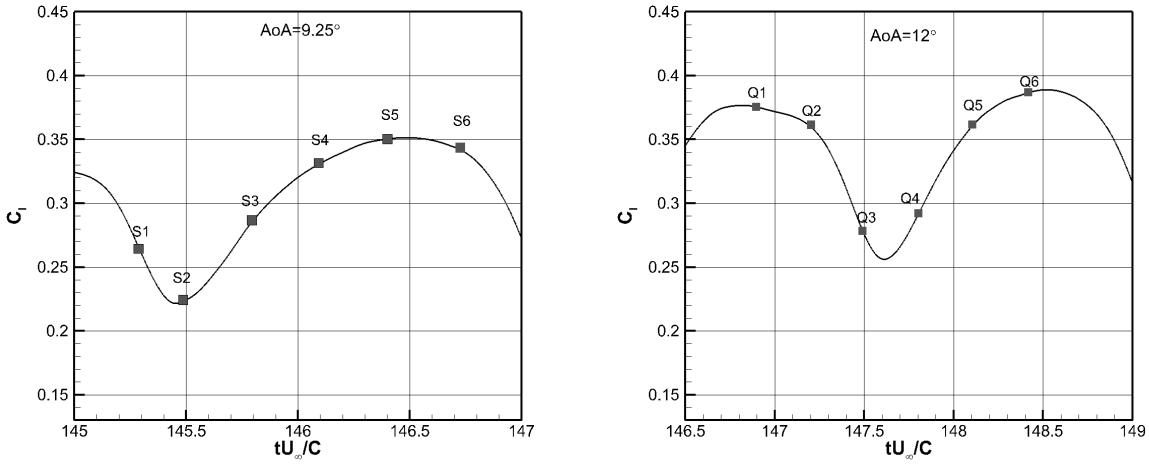
(a) **(b)**

Figure 6.6: The drag coefficient versus time at: (a) $AoA = 9.25°$; (b) $AoA = 12°$.

6.1.4 Instantaneous flow field

Pressure contours

The pressure contours are plotted in Figure 6.8 at the mid-span plain for $AoA = 9.25°$ for times S1 to S6 on the lift coefficient plot shown in 6.7. Pressure is non-dimensionlized using the velocity and pressure inlet $(C_p = (p - p_\infty)/(0.5\rho U_\infty))$. The flow is initially laminar. However, due to rapid changes of the shear stresses inside the boundary layer on the suction side, the Kelvin-Helmholtz



(a) **(b)**

Figure 6.7: Lift coefficient variation over one cycle at: (a) $AoA = 9.25°$; (b) $AoA = 12°$.

instability appears. This instability results in building up the roller vortex, and perturb the laminar boundary layer. Eventually it makes the flow completely turbulent. Some vortices that are created on the suction side travel to the trailing edge and Von Karman vortices shed at the trailing edge and travels downstream. At time S5 the largest vortex appears on the suction surface, which is in agreement with the high lift coefficient that is observed in Figure 6.7. Vortices create a low pressure region, therefore a largest vortex creates the biggest pressure difference between two sides of blade. Consequently, S5 has the highest lift coefficient. On the contrary, at time S2 Von Karman vortices have just left the surface. Therefore, the minimum value of lift is observed at this point.

Similarly, Figure 6.9 displays the pressure contours at the $AoA = 12°$ for the times Q1 to Q6 (Figure 6.7). At this angle, the instability appears closer to the leading edge, and the vortices are stronger in comparison with $AOA = 9.25°$. This confirms the higher average lift coefficient at $AOA = 12°$. The Von Karman vortices shed to the free-stream at the trailing edge of the blade. At times Q1 and Q5, the flow experiences the largest vortices, correspondingly the highest lift at these points. On the other hand, the lowest lift, among the marked points, appears at time Q3 where the vortex has left the blade.

Velocity vectors

Figures 6.10 and 6.15 present the velocity vectors and their magnitude around the blade at the mid-span plain for the both AoAs. It is observed that the magnitude and direction of velocity changes severally on the section side. However, the flow always remained attached to the surface on the pressure side. This means that we could refine the mesh even finer on the suction side, and coarsen on the pressure side without significant compensation on the CPU time. This also can be implemented to the CFD codes that have adaptive feature by selecting the velocity derivative as an adaptive criterion. This is due to having the large velocity variations in the separated region.

It is seen that at the leading edge flow is separated from the surface, and a laminar bubble emerges almost for all 12 states. However, their size and location change, as seen in Figures 6.10f and 6.11f. The separation at all the states S1 to S6 for $AoA = 9.25^{\circ}$ appears at $(x \approx 0.22/C)$, while it moves upstream $(x \approx 0.12/C)$ for the $AoA = 12^{\circ}$.

A close-up of the flow at the leading edge is shown in Figure 6.12 for S5 and Q1, where the maximum lift has been observe. It is seen that, not only separation happens earlier $AoA = 12^{\circ}$ in this case, it also has a thicker inverse flow boundary layer. In Figure 6.13 the tailing edge are shown at the same points. The difference between the boundary layer on the suction and pressure side is apparent. On the pressure side an attached laminar boundary layer is observed. while on the section side the flow is completely influenced by the separation, vortices and coherent turbulent structures.

Vorticity contours

More details about movement and size of the vortices are observed by looking at the vorticity contours (Figure 6.14 and 6.15). The higher vorticity can be interpreted as a large region of the rotational flow. The rotational flow is very common in viscous flows, specially in the boundary layer where an inhomogeneous normal derivative of the velocity component is observe. For example at the sub-layer of the boundary layer, the streamwise velocity variation in normal to the surface, $(\partial u_s/\partial x_s)$, is orders of magnitude larger than the cross-wise velocity variation ,$(\partial u_n/\partial x_n)$, where

subscripts "n" and "s" represent the normal- and stream-wise directions. Therefore, the maximum of this value happens in a thin layer near the wall. High values are also observed in laminar separated boundary region, as well as inside a viscous vortex. It is seen that after transition from the laminar to turbulent, vortices are rolled-up continuously and transfered to the downstream. A pattern is shown in Figures 6.15e and 6.14f for $AoA = 9.25°$, where two clock-wise vortices are generated on the suction surface and trapped a Λ-shape structures. This behavior is more obvious at $AoA = 12°$, due to stronger vortices that exist near the suction surface (Figures 6.15a and 6.15b).

Coherent turbulent structures

Flow transition from the laminar to turbulent is considered a full 3-dimensional phenomena. The Kelvin-Helmholtz (KH) instability is generated due to the velocity gradient and the large shear stresses inside the boundary layer. Herein a parameter is defined in order to better represent of turbulent structures, $\mathcal{Q}$-criterion is defined as the difference of vorticity tensor and strain-rate tensor. The former is the skew-symmetric part of the velocity gradient, while the later is symmetric component, $\mathcal{Q}$-criterion is given by,

$$\mathcal{Q} = \frac{1}{2}(||\mathbf{\Omega}|| - ||\mathbf{S}||), \tag{6.3}$$

where $||\mathbf{\Omega}||$ and $||\mathbf{S}||$ are the Euclidean norm of the vorticity and the stain-rate, respectively. The iso-surface of $\mathcal{Q} = 30$ is shown in Figure 6.16 and 6.17. Instability is introduced to the flow by some oscillations at the edge of the laminar boundary layer (BL) (Figure 6.16f, 6.17a and 6.17b). Then, a roll-up vortex is built inside the transitional region, and eventually the flow breaks up completely and smaller turbulent structures are formed.

The coherent turbulent structures can be seen by the iso-surface of the pressure gradient (Figures 6.19 and 6.18). A Hairpin vortex is usually generated after transition from the laminar to turbulent boundary layer due to the KH instability. A Hairpin vortex is built up from two counter-rotating vortex legs along the stream-wise direction and a span-wise head-vortex. Some Hairpin vortices are highlighted in Figures 6.18e, 6.19a, and 6.19f. These observations confirm that the transition procedure is an entire 3-dimensional phenomena, and it is not possible to predict this with 2-dimensional

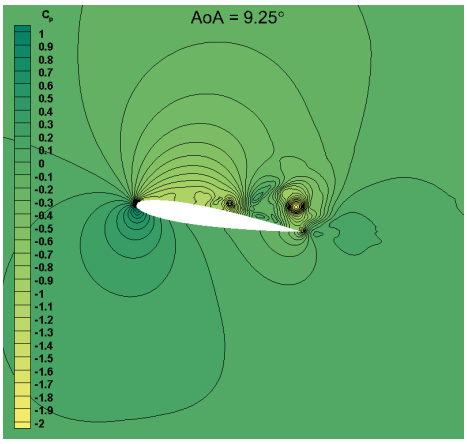
URANS simulations.

Instantaneous velocity plots

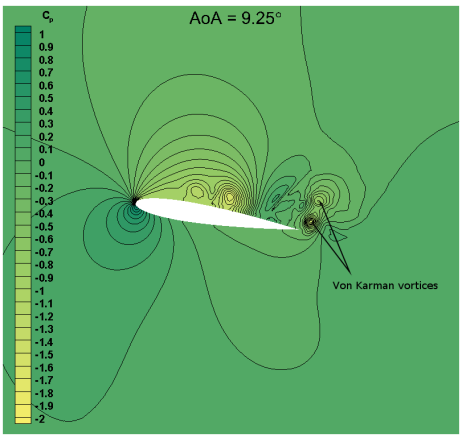
In this sections, 5 points are located near the blade's surface and inside the wake (Figure 6.20). The variation of velocity over time is investigated at these points in Figure 6.21. At location P1, flow is laminar, and can even be considered steady. This point is located outside of the boundary layer. Then at point P2, flow begins the transition. However, it is not completed yet. At P3, flow is completely turbulent for both $AoA = 9.25°$, and $AoA = 12°$. The velocity at points P4 and P5 changes according to the Von Karman vortex frequency.

6.1.5 Mean Pressure Distribution

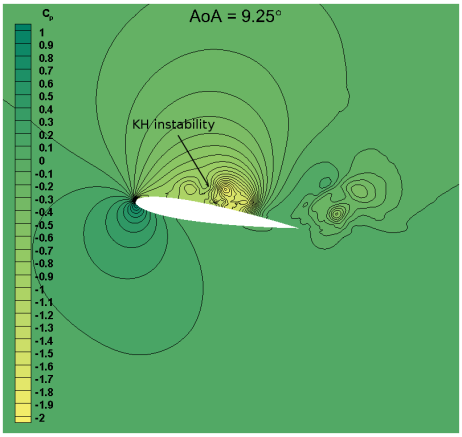
The average of the pressure coefficient obtained over 10 pseudo-cycles, see Figure 6.7. The Pressure is demonstrated in Figure 6.22. It is seen that the mean pressure is higher for the $AoA = 12°$, The plateau for the pressure represent the separation of flow, and the pressure after reattachment start increasing again. The inverse flow may cause to the pressure drops at $AoA = 12°$.



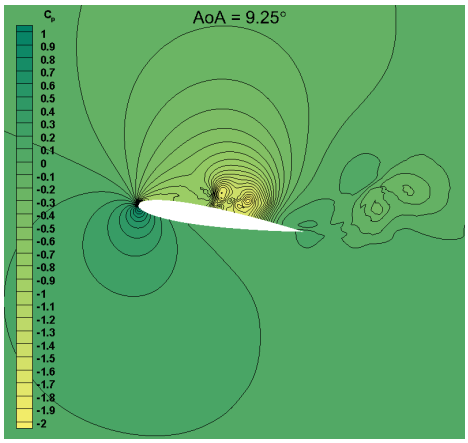
(a) S1



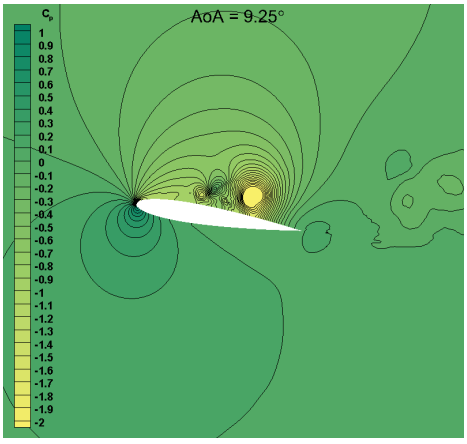
(b) S2



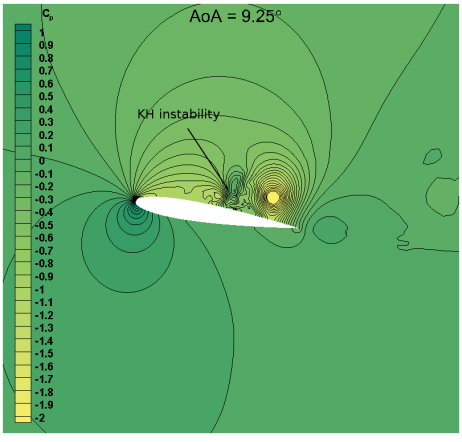
(c) S3



(d) S4

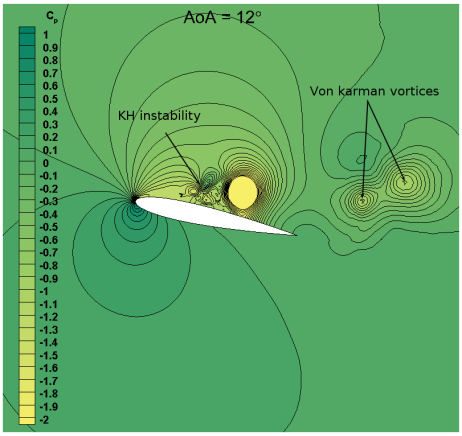


(e) S5

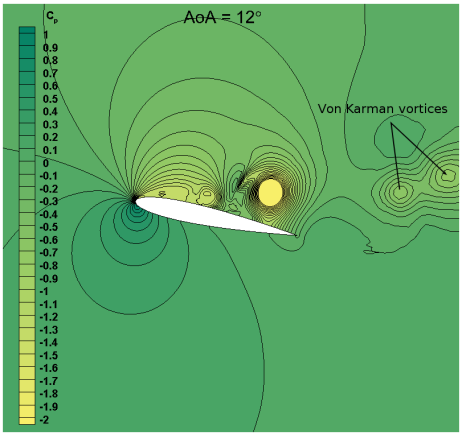


(f) S6

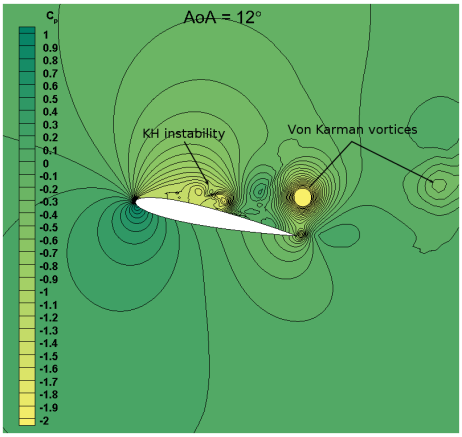
Figure 6.8: Instantaneous pressure contours $(AoA = 9.25°)$.



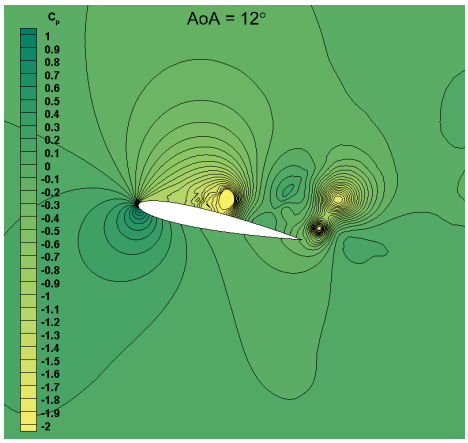
(a) Q1



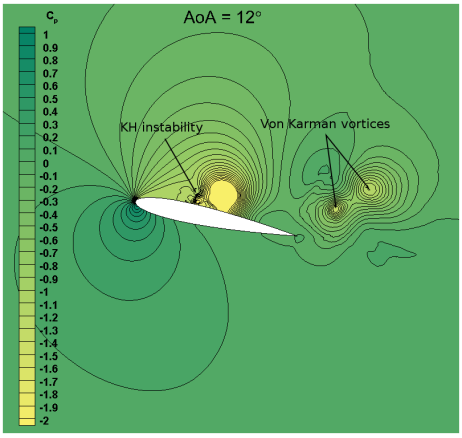
(b) Q2



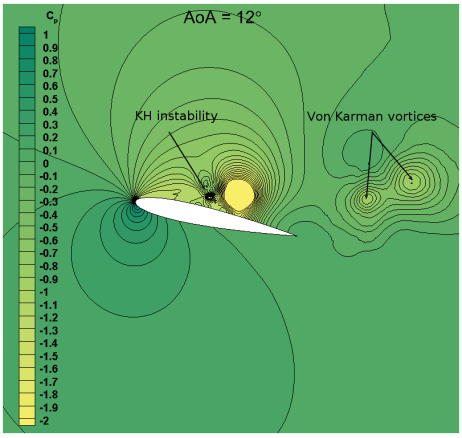
(c) Q3



(d) Q4

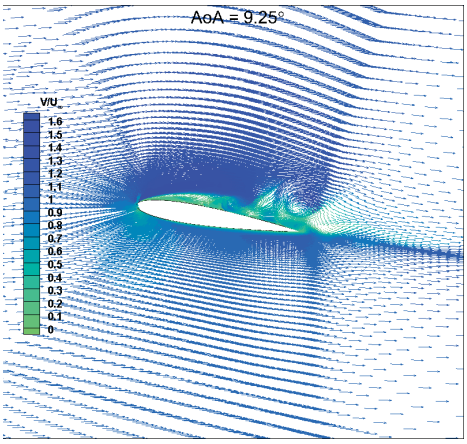


(e) Q5

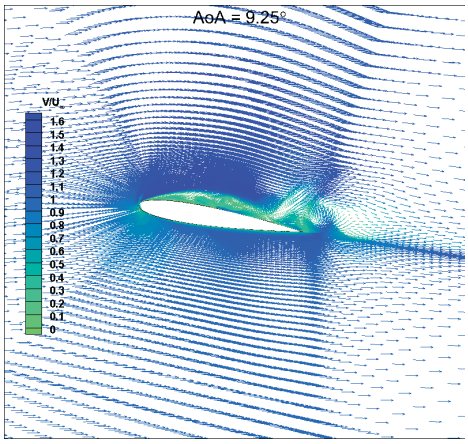


(f) Q6

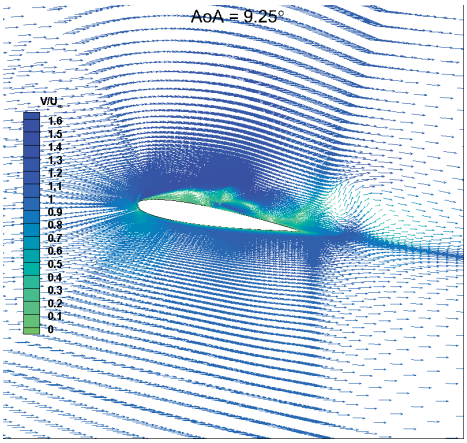
Figure 6.9: Instantaneous pressure contours $(AoA = 12°)$.



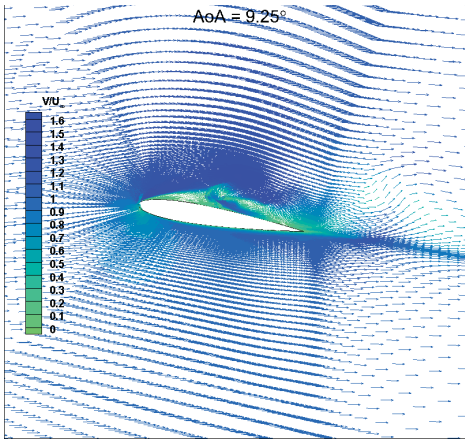
(a) S1



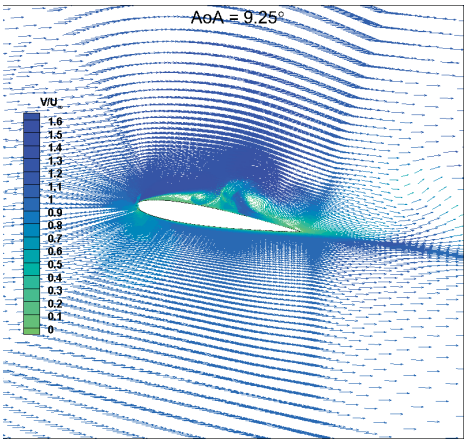
(b) S2



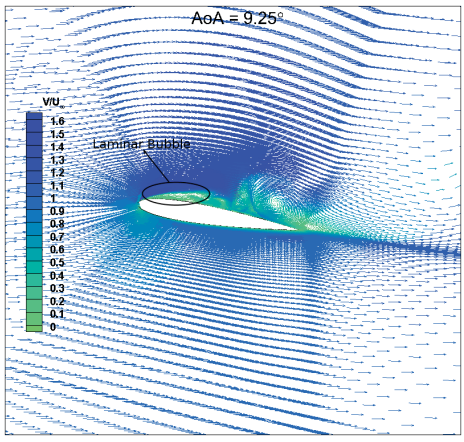
(c) S3



(d) S4

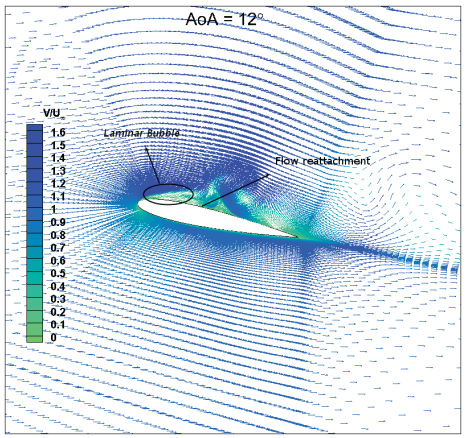


(e) S5

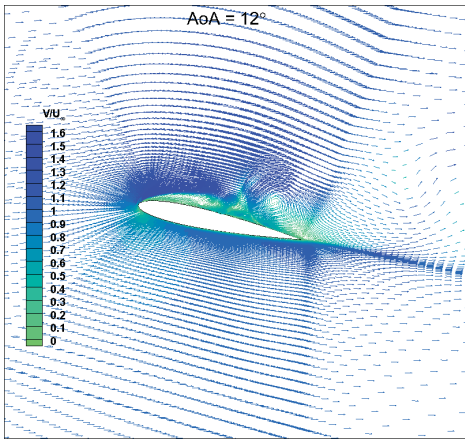


(f) S6

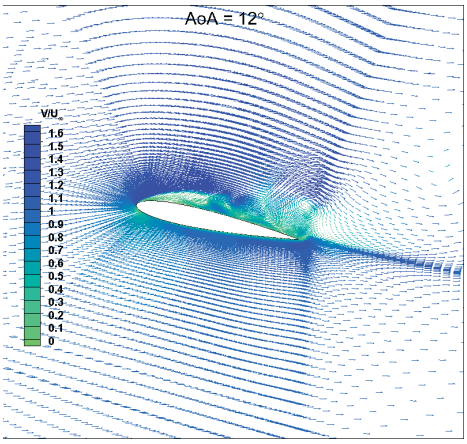
Figure 6.10: Instantaneous velocity vectors $(AoA = 9.25°)$.



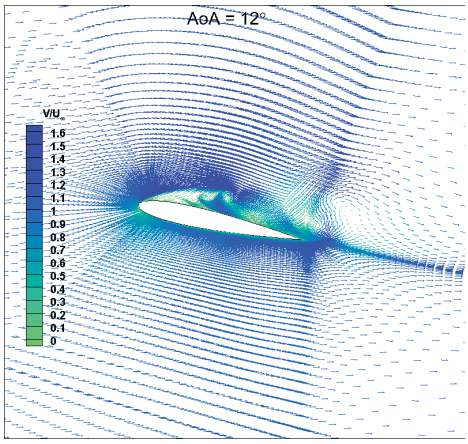
(a) Q1



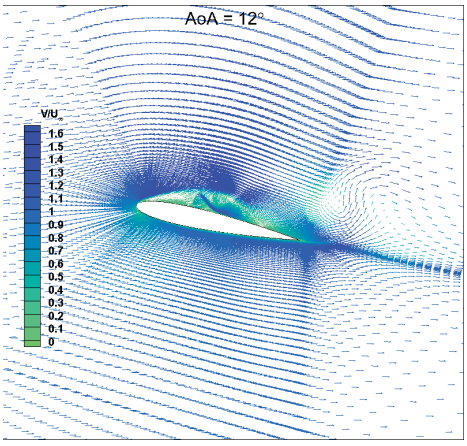
(b) Q2



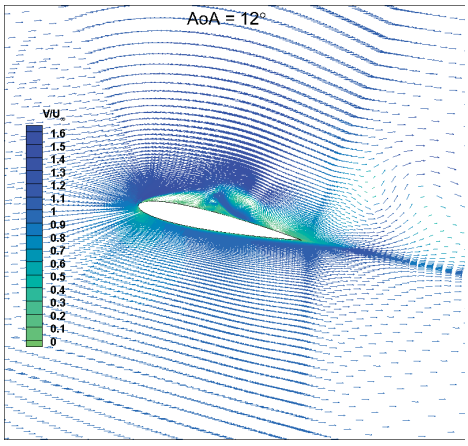
(c) Q3



(d) Q4

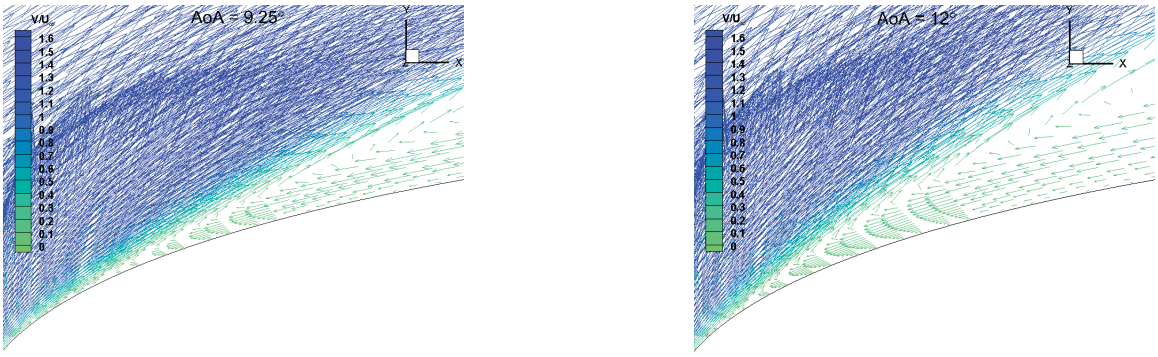


(e) Q5



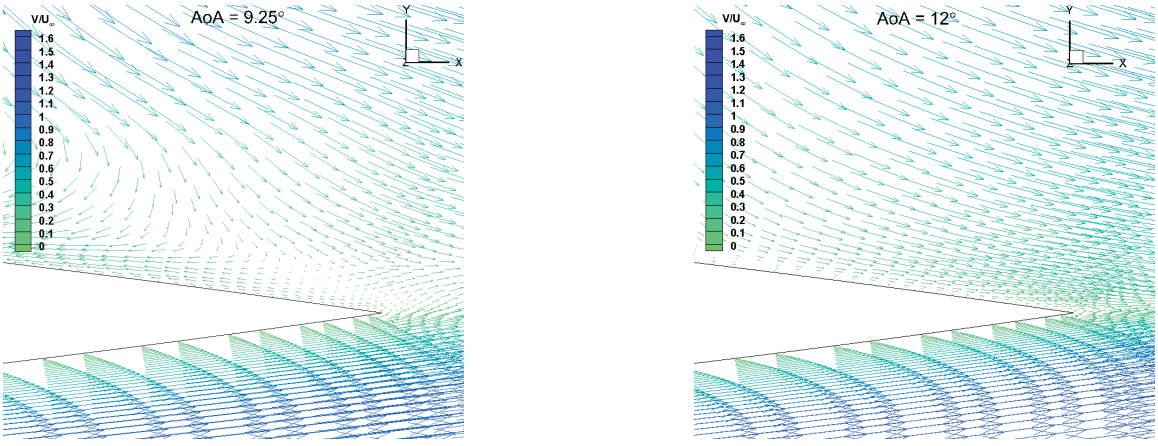
(f) Q6

Figure 6.11: Instantaneous velocity vectors $(AoA = 12°)$.



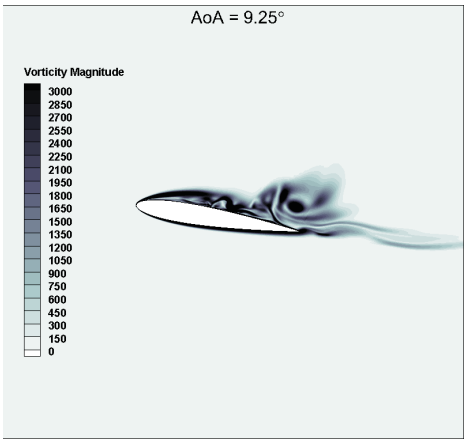
(a) **(b)**

Figure 6.12: Instantaneous velocity vectors at the leading edge. (a) $AoA = 9.25°$, S5; (b) $AoA = 12°$, Q1.

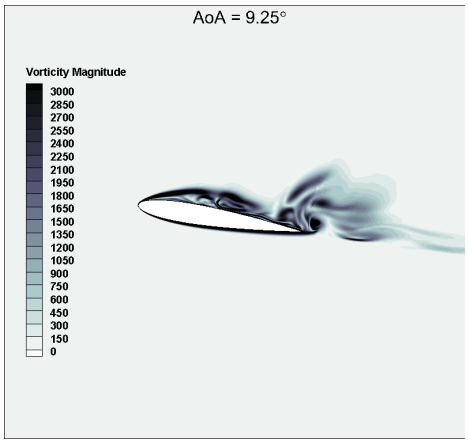


(a) **(b)**

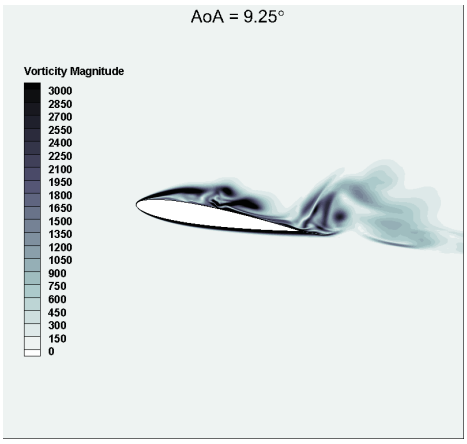
Figure 6.13: Instantaneous velocity vectors at the trailing edge. (a) $AoA = 9.25°$, S5; (b) $AoA = 12°$, Q1.



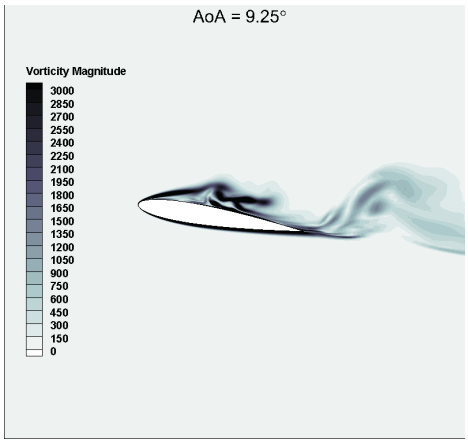
(a) S1



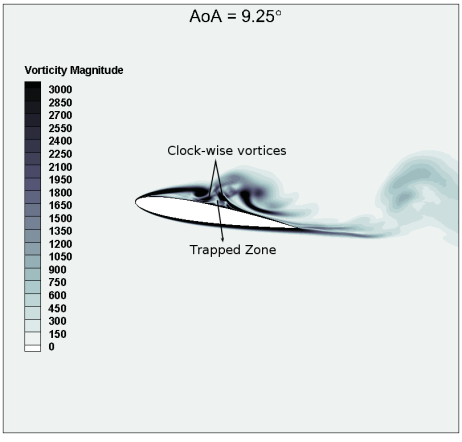
(b) S2



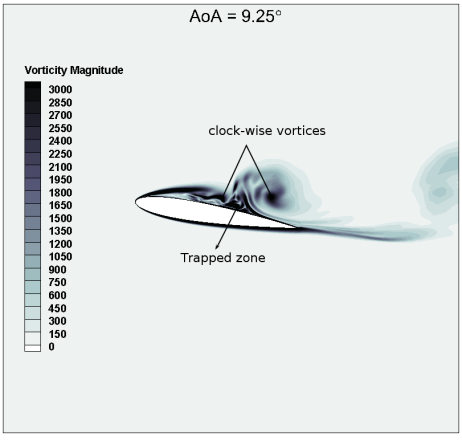
(c) S3



(d) S4

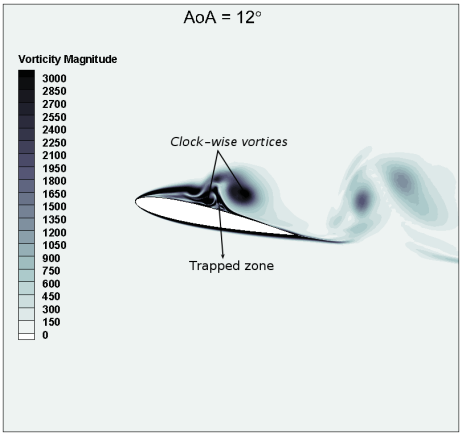


(e) S5

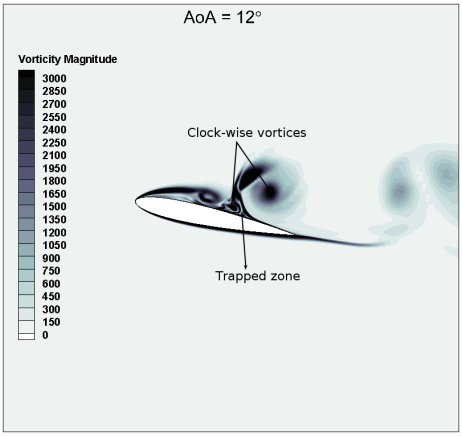


(f) S6

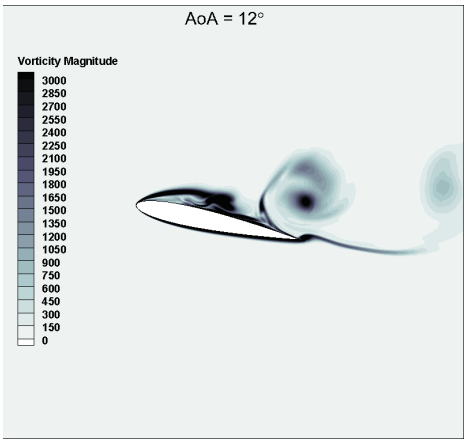
Figure 6.14: Instantaneous vorticity contours $(AoA = 9.25°)$ [1/s].



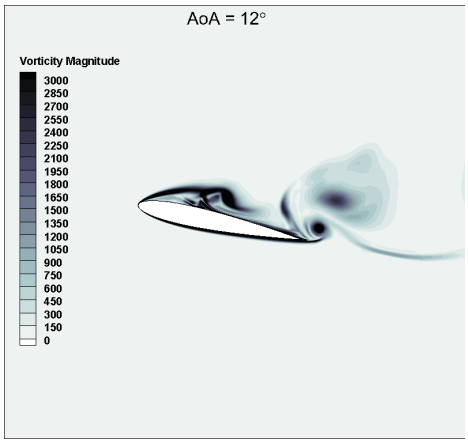
(a) Q1



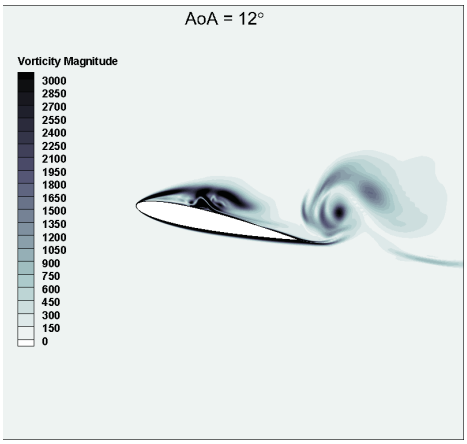
(b) Q2



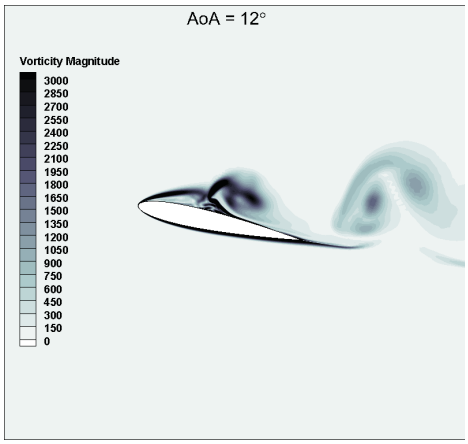
(c) Q3



(d) Q4

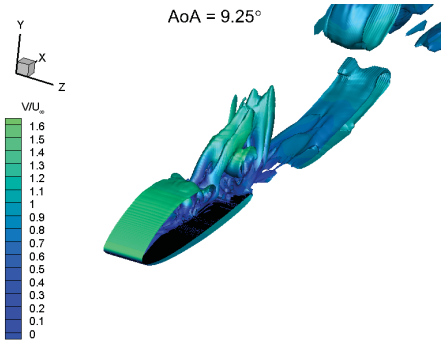


(e) Q5

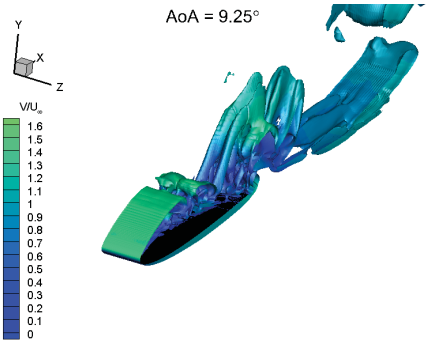


(f) Q6

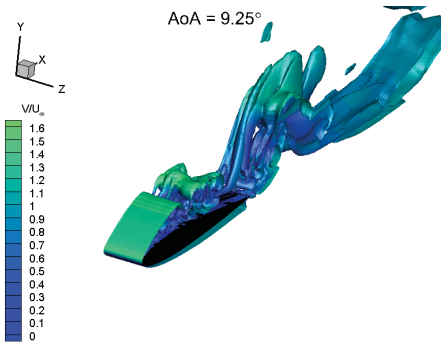
Figure 6.15: Instantaneous vorticity contours $(AoA = 12°)$ [1/s].



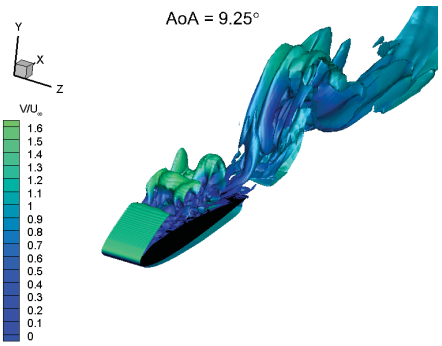
(a) S1



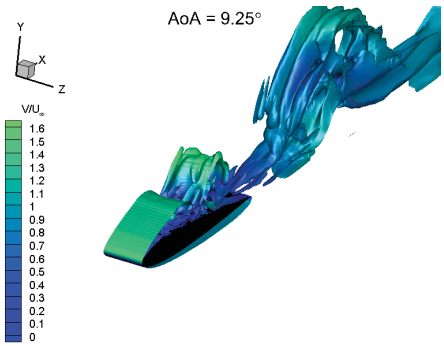
(b) S2



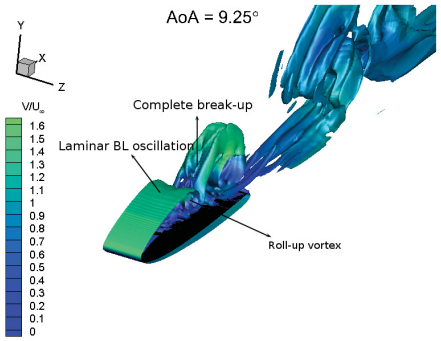
(c) S3



(d) S4

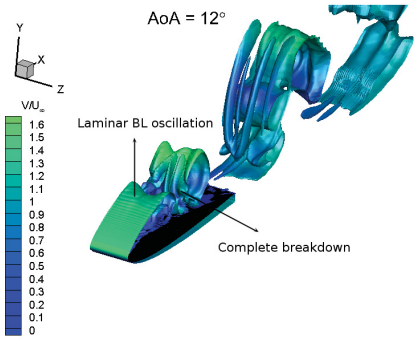


(e) S5

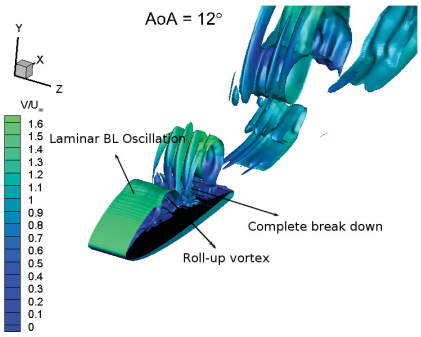


(f) S6

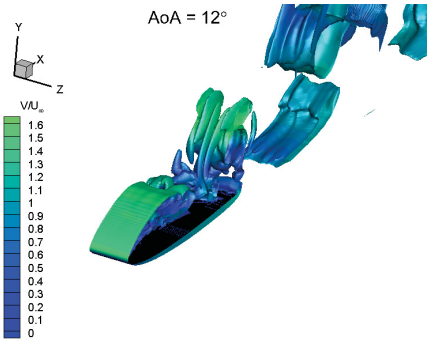
Figure 6.16: Instantaneous iso-surface of the $\mathcal{Q} = 30$ with the contour of the velocity magnitude ($AoA = 9.25°$).



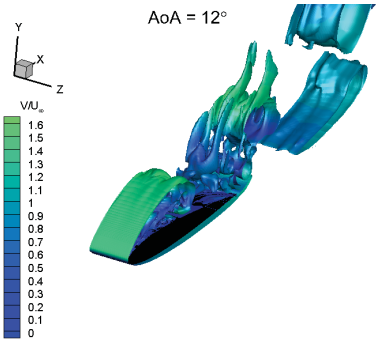
(a) Q1



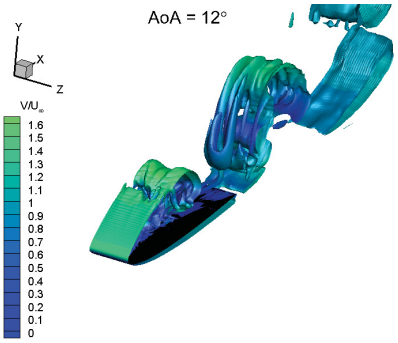
(b) Q2



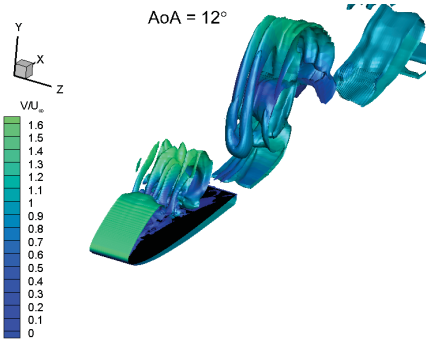
(c) Q3



(d) Q4

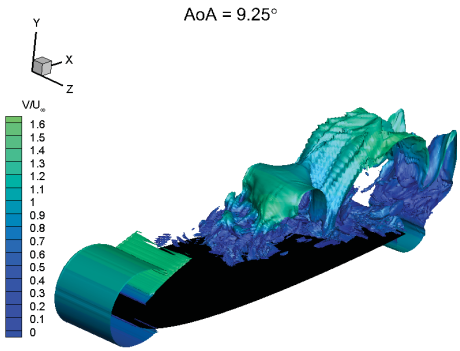


(e) Q5

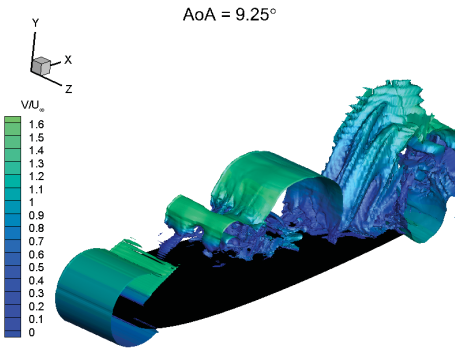


(f) Q6

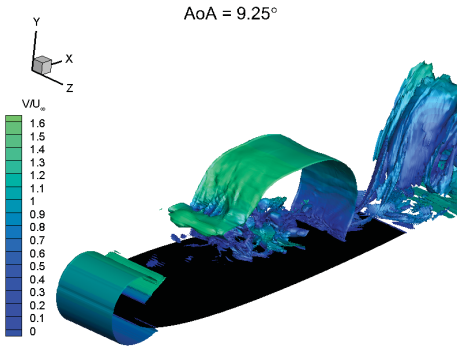
Figure 6.17: Instantaneous iso-surface of the $\mathcal{Q} = 30$ with the contour of the velocity magnitude ($AoA = 12°$).



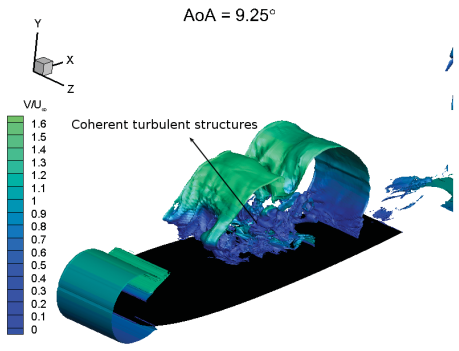
(a) S1



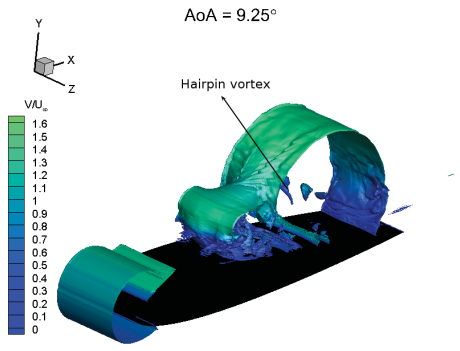
(b) S2



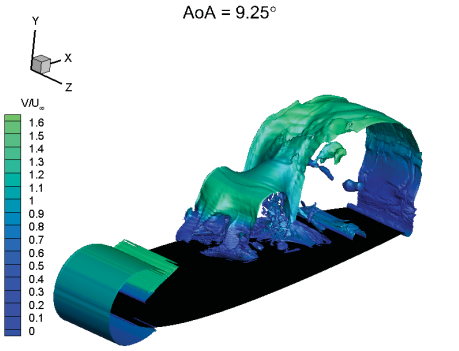
(c) S3



(d) S4

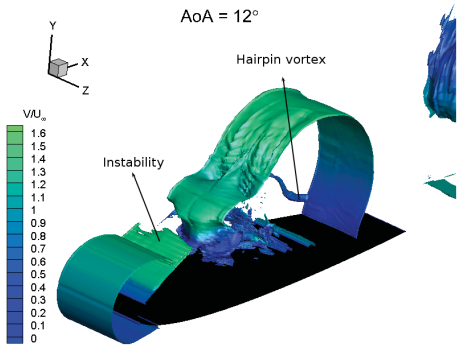


(e) S5

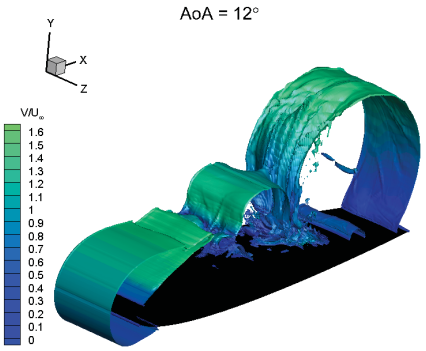


(f) S6

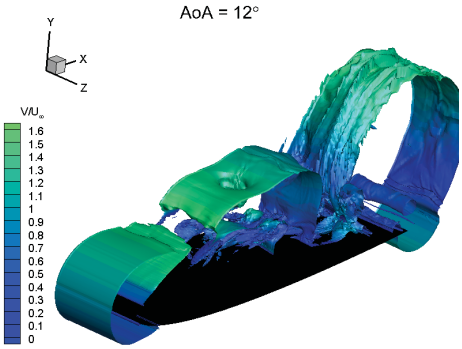
Figure 6.18: Instantaneous iso-surface of pressure gradient $|\partial p/\partial x| = 20000[Pa/m]$ with the contour of the velocity magnitude $(AoA = 9.25°)$.



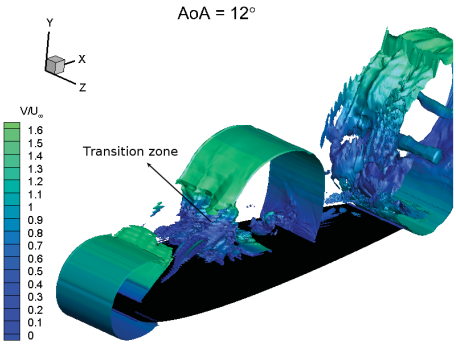
(a) Q1



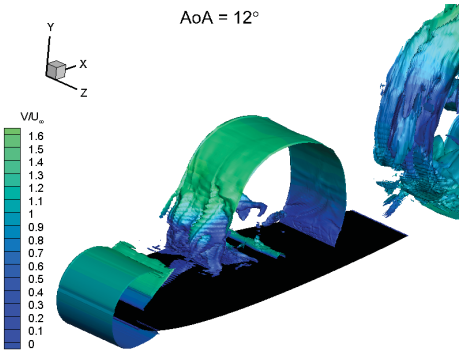
(b) Q2



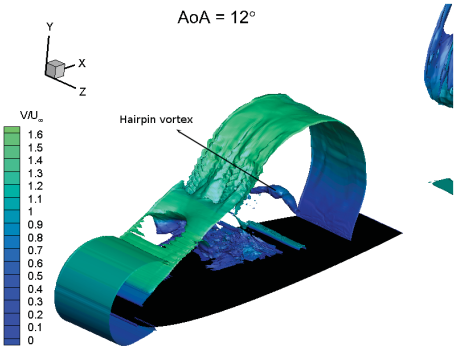
(c) Q3



(d) Q4



(e) Q5



(f) Q6

Figure 6.19: Instantaneous iso-surface of pressure gradient $|\partial p/\partial x| = 20000[Pa/m]$ with the contour of the velocity magnitude $(AoA = 12°)$.

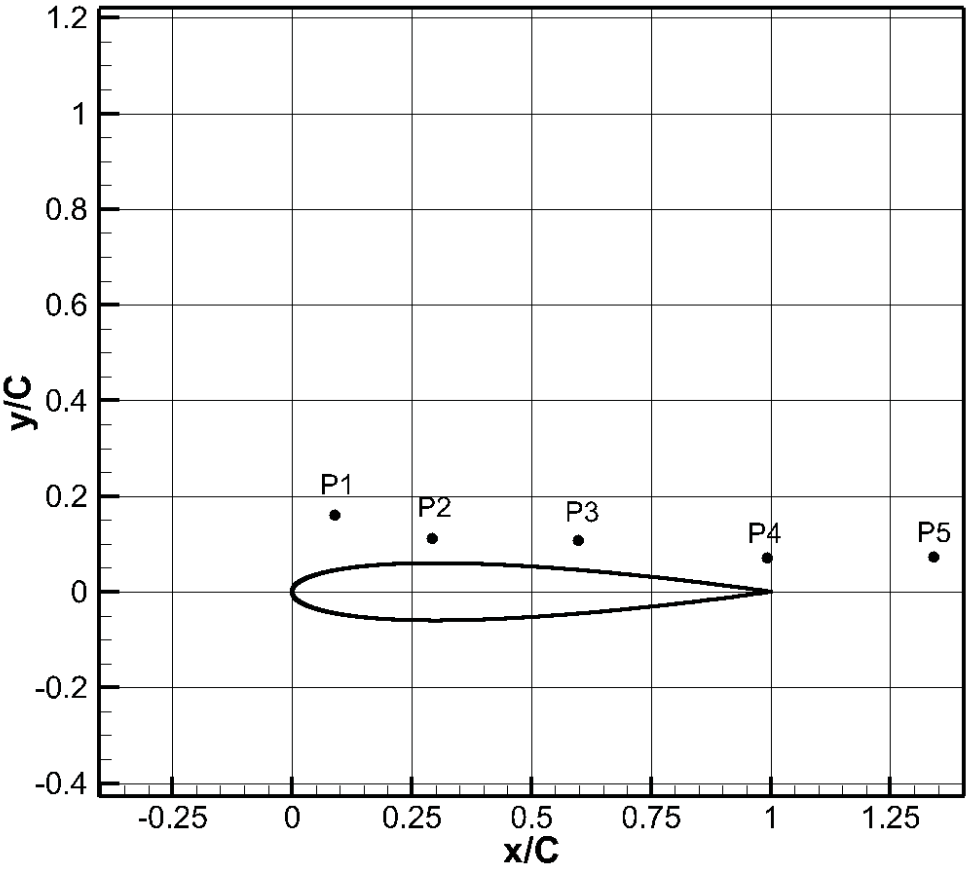
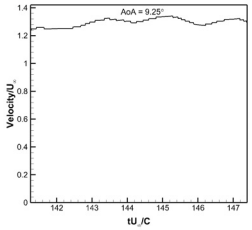
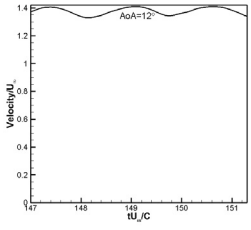


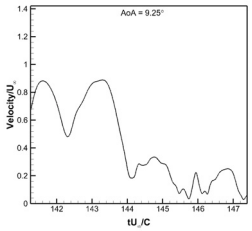
Figure 6.20: Location of poits around the NACA 0012 Profile.



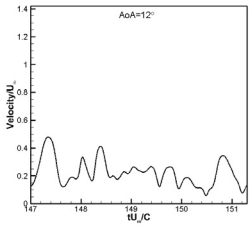
(a) P1



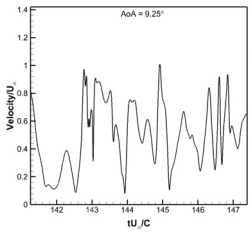
(b) P1



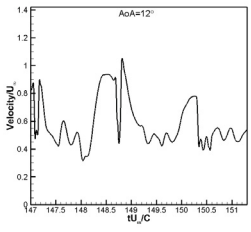
(c) P2



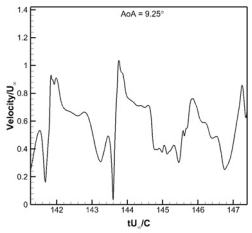
(d) P2



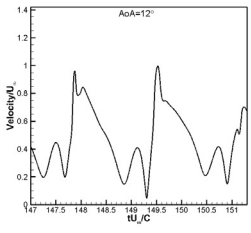
(e) P3



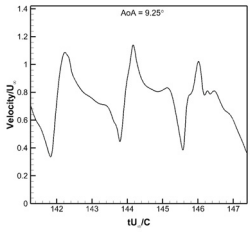
(f) P3



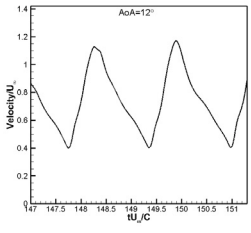
(g) P4



(h) P4

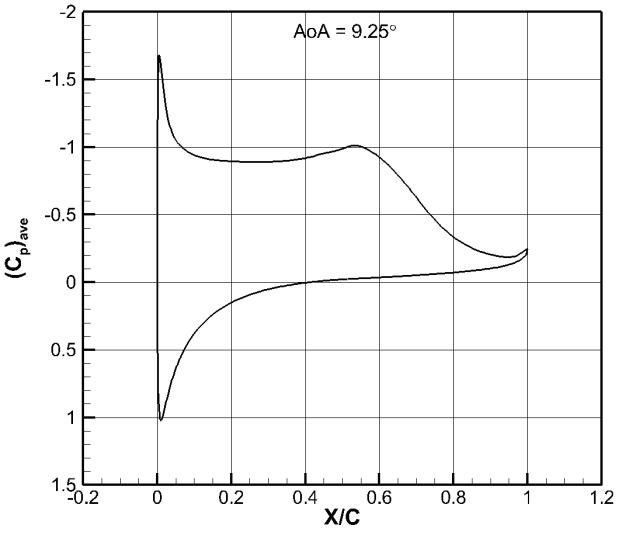


(i) P5

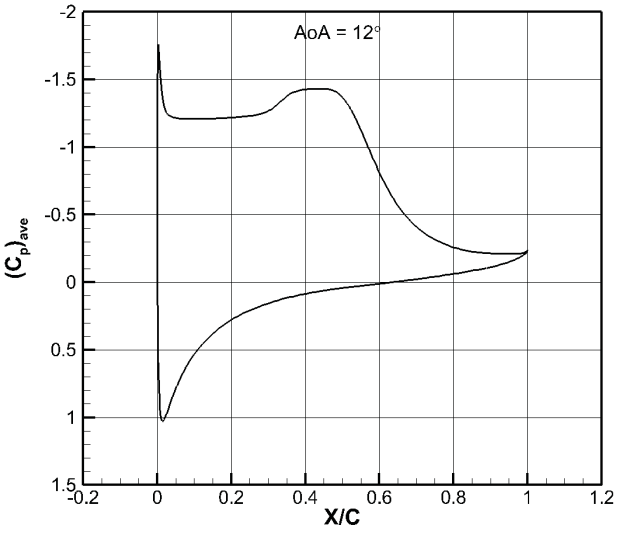


(j) P5

Figure 6.21: Velocity fluctuation in time at points P1, P2, P3, P4, P5.



(a)



(b)

Figure 6.22: Mean pressure coefficient. (a) $AoA = 9.25°$; (b) $AoA = 12°$.

6.2 Fixed-Blade Vertical Axis Wind Turbine on Mars

A H-Blade wind turbine is simulated with a rigid blade in the current section. The blade is chosen based on the work done by the Pankonien [86] under the supervision of Prof. Inman at the University of Michigan. They introduced a new way of creating a Spanwise Morphing Trailing Edge (SMTE) wing, which is proposed as an effective model to optimize the aerodynamic loads on the airplane's wings. Therefore, it can be an alternative to the flaps and slots. They combined Macro-Fiber Composites (MFCs) with a flexure box mechanism to create the SMTE wing for an Unmanned Aerial Vehicle (UAV) (Figure 6.23). They used a multi-material 3D printer to print the flexure box from the combination of elastometric materials and rigid plastics. They showed that this method can improve the smoothness of the tailing edge in comparing with other morphing technique such as hinged box model [85]. To morph the blade, two actuators were utilized in order to apply a voltage in the range of -2kV to 2kV to deform the smart materials.

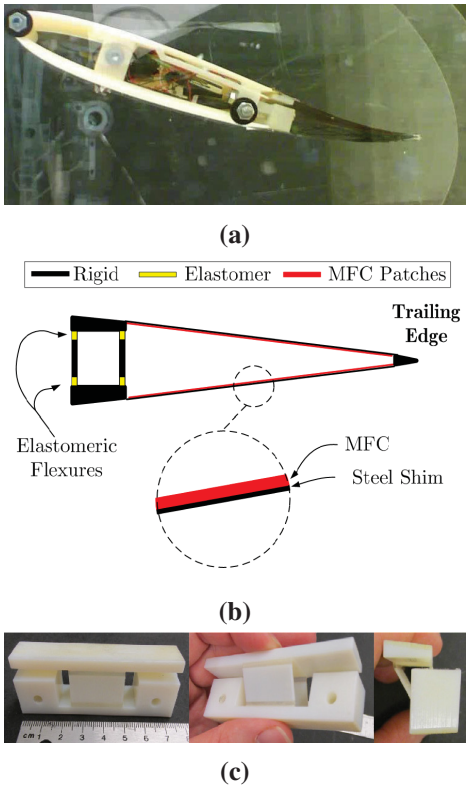


Figure 6.23: A Morphing blade with the flexure box and Macro-Fiber composite materials. (a) Tip deflection at $V = 20[m/s]$ and $\alpha = 20°$; (b) Diagram of flexure box; (c) Flexure box construction [86].

6.2.1 Mesh and boundary conditions set-up

The NACA 0012 is the based airfoil profile proposed by Pankonien and Inman. The chord length is $C = 0.305(m)$. The profile is rigid from the leading edge to $x/C = 0.52$, then it starts to deform in the second half of the NACA 0012 profile. They provided us with 21 different airfoil profiles. A Fourier curve fitting series, with 6 modes, is used to find the equation of the upper and lower profile, separately. It is is given by,

$$y(x) = \sum_{n=0}^{6} (a_n \cos(nfx) + b_n \sin(nfx)), \tag{6.4}$$

where frequency, f, and the constants, a_n and b_n, are evaluated with the available points on the blade over the deformed segments. Our objective in this section is to analyze the flow with a fixed profile. Therefore, we select three profiles out of 21 to use in a VAWT: one which the closest to the symmetric NACA 0012 profile (FR14), and two of the most extreme chambered profiles (FR1 and FR21) (Figure 6.24). The Fourier's constants corresponding with these three profiles are presented in Appendix C.

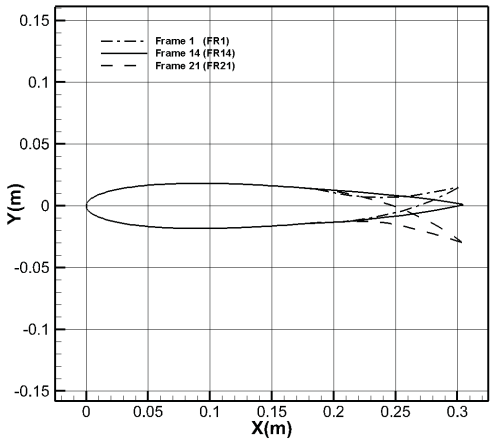


Figure 6.24: NACA 0012 airfoil profile and morphed airfoils.

The VAWT operates at the averaged atmospheric conditions on the surface of Mars (see Table 1.1). The turbine is tested at tip speed ratio ($R\omega/U_\infty$) of 3.8 , solidity number ($\sigma = N_b C/R$) of 0.1, and number of blades, N_b, is one in our simulation (Table 6.3). The 2.5-D domain is built by

extending the domain by 10% of the chord in the span-wise direction.

Table 6.3: The VAWT Geometry and the flow condition

Angular Velocity [rad/s]	Chord [m]	Number of Blades	Solidity	span/Chord	TSR
26.23	0.305	1	0.1	0.1	3.8

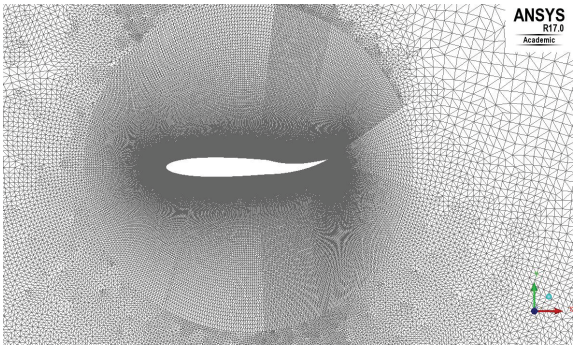
The Reynolds number for VAWTs is calculated based on the rotational speed of blade ($Re = \rho R\omega C/\mu$) instead of free-stream velocity. Therefore $Re = 24400$ and $Ma = 0.35$ are obtained on the Mars for this VAWT. However, the relative value of velocity is not constant, due to rotation of blade. The maximum and minimum Reynolds number that blade experiences based on the relative velocity are 30000 and 18000, respectively.

The mesh generation strategy is similar to the one described in the previous section for the steady blade. A structured mesh is generated close to the blade profile (see circle region in Figure 6.25). However, an unstructured grid is created initially for the outer part of domain (see Figure 6.26). A closer look at the mesh on the surface of the blade is shown in Figure 6.27. More details about the mesh quality and size are summarized in Table 6.4. Y^+ in this case constantly changes. However, with the previous analysis on the single blade, the first height distance (Δy_1) is chosen in the way that it guarantees $Y^+ < 1$ for the solution.

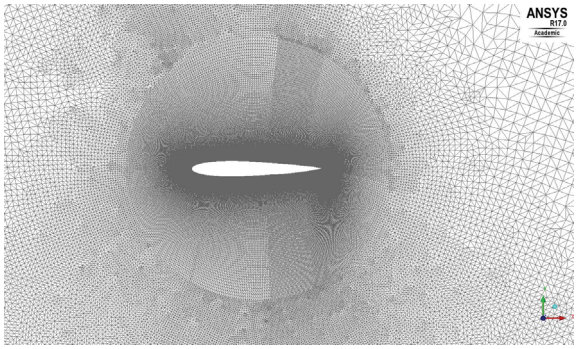
Table 6.4: Mesh information summary for the VAWT

Cell Type	Tetrahedral
Number of Cells	2.5×10^7
Number of Nodes	4.5×10^6
$\Delta y_1/C$	0.0005
Geometric Growth Ratio	1.1
x^+/y^+	10
z^+/y^+	10

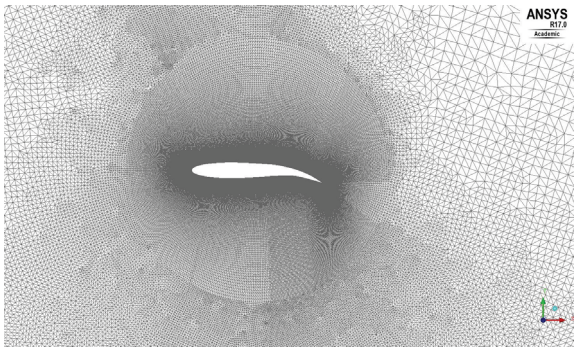
The flow enters the domain from left with a zero angle with respect to the x-axis, a periodic boundary condition is set in the span-wise boundary faces and the pressure far-field at the outer domain which is placed at $10 \times R$. Finally, the no-slip wall condition is chosen on the surface of blade.



(a) FR1



(b) FR14



(c) FR21

Figure 6.25: Structured region close to the blade surface. (a) Morphed outward; (b) Close to NACA 0012 ; (c) Morphed inward.

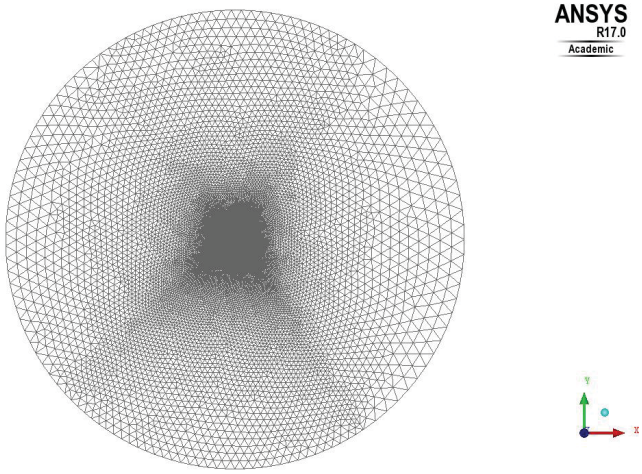
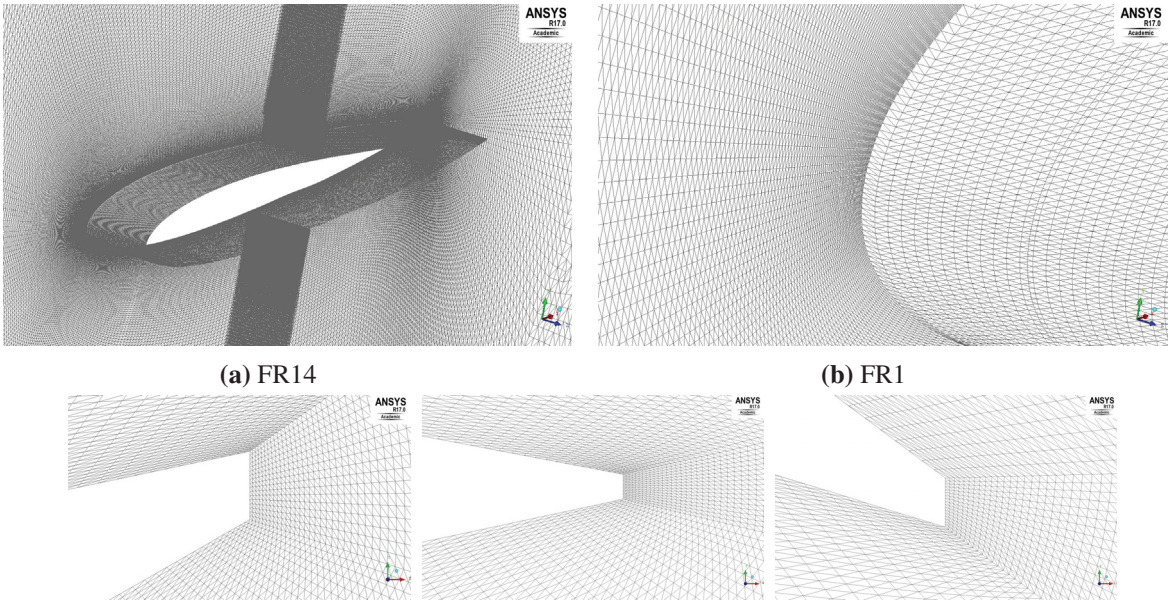


Figure 6.26: Unstructured mesh around the Far-field for VAWT.



(a) FR14 **(b)** FR1

(c) FR1 **(d)** FR14 **(e)** FR21

Figure 6.27: Mesh quality on the blade surface. (a) The Blade surface (b) Leading Edge; (c) Trailing edge FR1 (d) Trailing edge FR14 (e) Trailing edge FR21.

6.2.2 The numerical set-up

Similar to the steady case, a second-order accurate implicit scheme is chosen in time and space. Then, the MUSCL scheme with the Van-Albada limiter is chosen, and the numerical dissipation is adjusted with the wiggle detector method. A Jacobian-based method is chosen to handle the derivative of fluxes. The entire domain rotates with the turbine angular velocity to mimic the wind turbine rotation. Consequently, there is no deformation occurring inside the domain. The Arbitrary Lagrangian Eulerian (ALE) method is used to preserve the second order accuracy of the scheme. The mesh movement also added to the boundary conditions. The CFL number increases from 1, and reaches a maximum value of 200. This corresponds to the time step ($\Delta t = 1.0 \times 10^{-5}$). The number of Newtons' loops (nbnewton) also eventually increases from two at the CFL=50 to nbnwton=10 at the CFL=200 to preserve convergence. The linear residual criteria inside GMRES is (10^{-6}). These values are also chosen based on the convergence study that is shown in the previous section. The WALE method is used as the turbulence model. The domain is decomposed into 288 processors with the software METIS, and it needs to run for at least one month for five cycles on Colosse and

Guillimn clusters of Compute Canada.

The scalability of the in-house code for simulation of VAWT is shown in Figure 6.28. The horizontal axis represents number of processors, and the vertical axis shows the execution time at each time step for CFL=10. The code has been tested for number of processors from 72 to 1152 with a geometric sequence factor of 2. The mesh size and the solvers parameters are the same for all the simulations. It is seen that the required time almost linearly reduces by increasing number of processors. Therefore, the code can be run on more than 1000 processors, if there are sufficient computing resources. In addition, it shows the capability of the in-house to simulate flows with higher Reynolds number with LES.

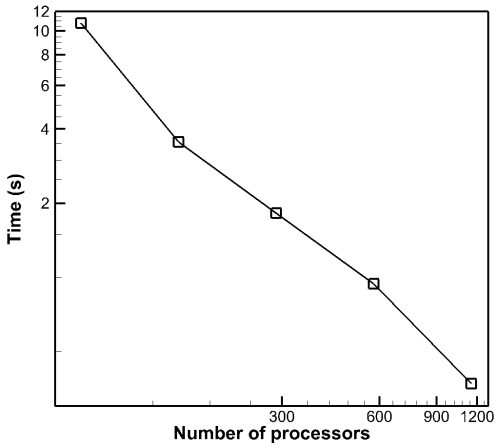


Figure 6.28: The in-house code scalability for VAWT simulations

6.2.3 Solution Analysis

In this section, the aerodynamic loads on the surface of the VAWT for three aforementioned morphed profiles are analyzed. First, the flow for the morphed inward profile (FR21) is shown. Then, a comparison between the aerodynamic loads of three profiles are presented to highlight the significance of blade shape on the performance of VAWTs. Frame 21 is chosen because the blade experience very high angle of attacks with the current conditions, and not because it might generate the maximum power.

The lift and drag coefficients versus the angle of attack and azimuthal position are shown in

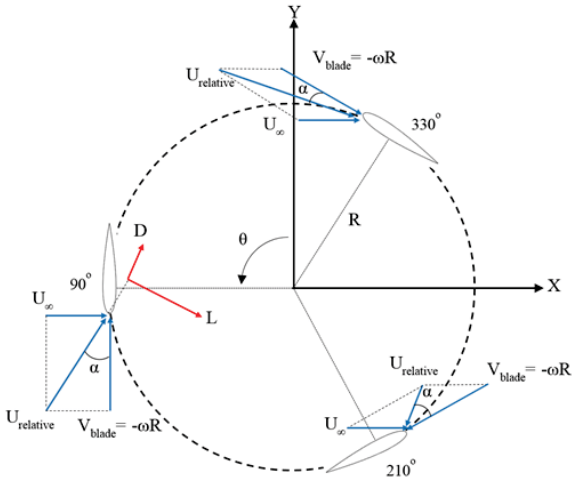
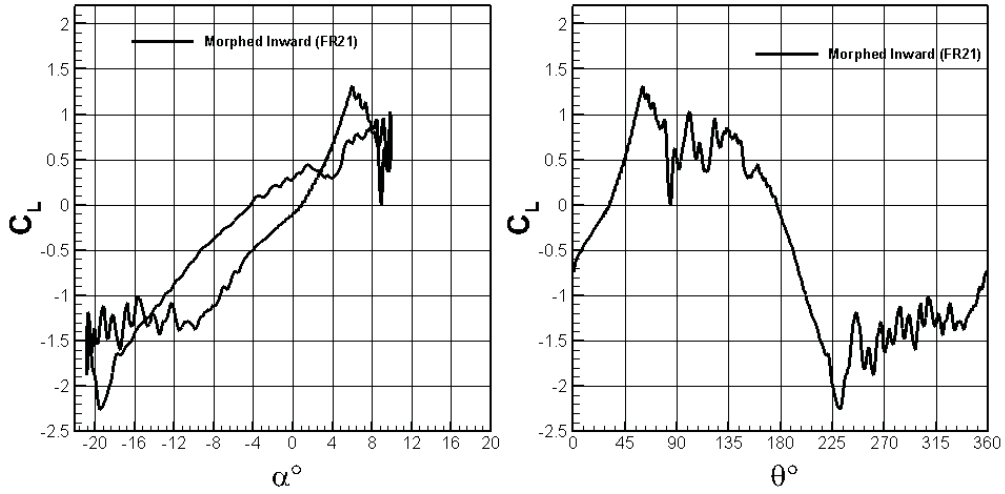


Figure 6.29: Angle of attack at different azimuth angle [40].

Figures 6.30 and 6.31, respectively. The AoA is not zero at zero azimuthal position, but it has a negative value of approximately -5 degrees. The negative lift at this point can be described by the vorticity contours shown in Figure 6.33. As the azimuthal angle reaches about 90 degrees, a sharp drop at lift coefficient is observed, which also is accompanied with the slightly drop of the drag coefficient. Afterwards the lift between 90 to 135 degrees fluctuates, although its average remains almost constant. This is due to the emerging and leaving of the vortices with different size and frequency. Some burst to the smaller vortices, while others leave the blade. Thereafter, the lift continuously reduces and drag increases at azimuthal angle in the range of 135 to 225 degrees, which is also seen in the contour of vorticity at 220 degrees. Finally, from 225 to 360 degrees, transition from laminar to the turbulent flow inside the boundary layer appears, and the flow travels to the downstream. The lift coefficient remains constant with a negative value, and the drag coefficient drops. Note that, this is not quite the typical behavior of a blade when rotating as in a VAWT, but in this case the blade analyzed is morphed inward.

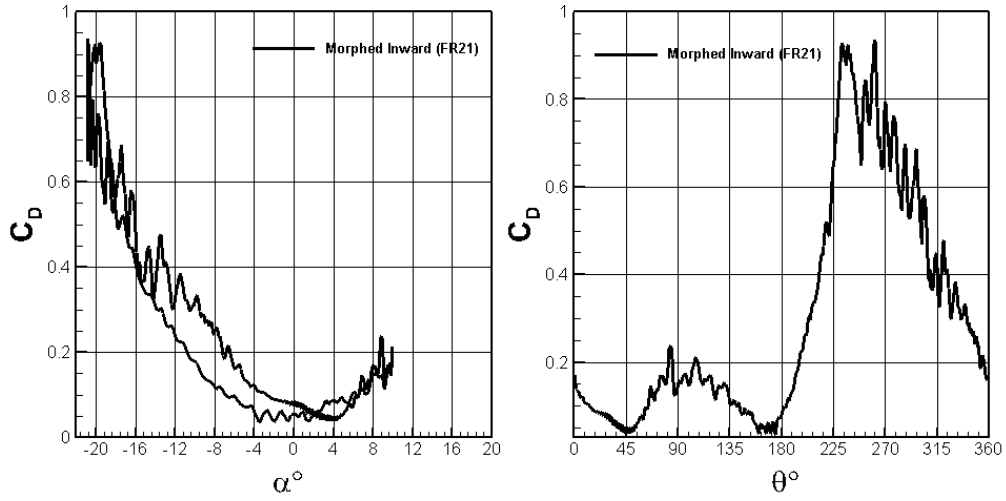
Figure 6.32 shows the power coefficient of turbine versus the azimuthal angle. The pattern of power usually follows the lift coefficient except those locations that the lift direction is toward the center of rotation such as the interval of 150 to 210. The average power for this turbine is -0.24. A negative power coefficient means, this turbine not only does not generate any significant power, but it needs to gain power to rotate (similar to a pump).

Finally, the wake behind the blade at the same azimuthal position (5, 90, 220, and 315 degrees) are illustrated in Figure 6.33. Notice that the cap value of vorticity is lowered from 5000 (1/s) in the previous case (near-wall zone) to the 100(1/s) for the sake of better illustration. It is seen that for almost at all the angles of attack the strake of vorticity is observed. The length and the strength of the wake is large on the second half of the turbine. It means that adding more two blades may have negative impact on the performance of the turbine.



(a) **(b)**

Figure 6.30: (a) The lift coefficient versus the angle of attack; (b) The lift coefficient versus the azimuthal angle.



(a) **(b)**

Figure 6.31: (a) The drag coefficient versus the angle of attack; (b) The drag coefficient versus the azimuthal angle.

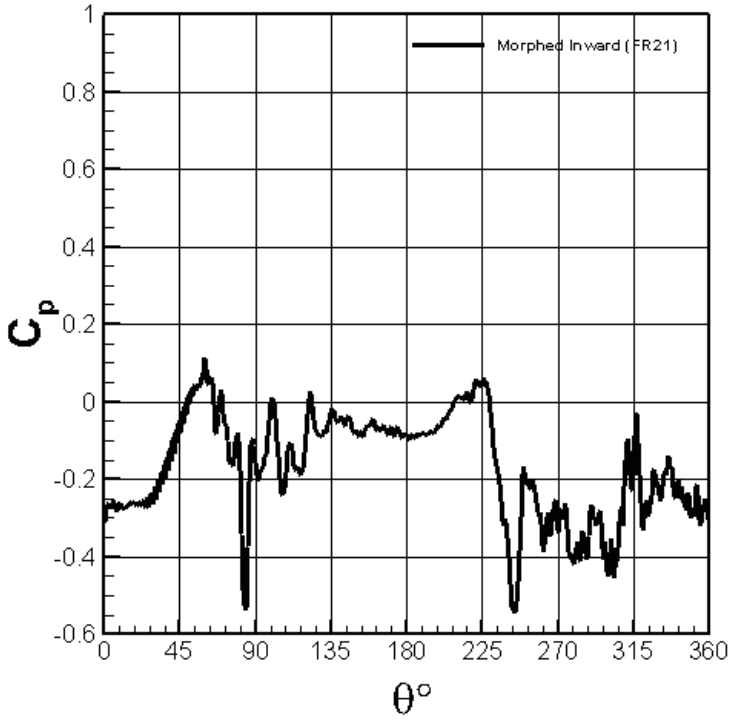


Figure 6.32: The power coefficient versus the azimuthal angel.

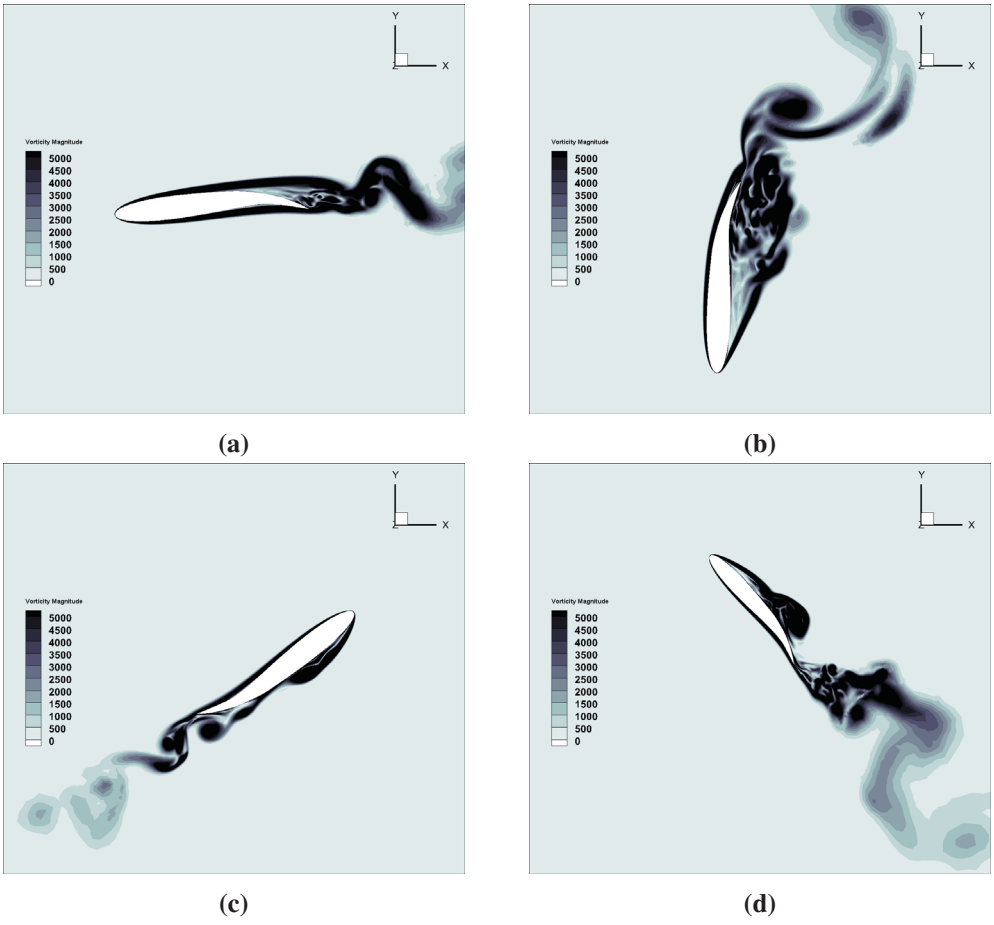


Figure 6.33: The vorticity contour near the blade [1/s]. (a) $\theta = 5°$; (b) $\theta = 85°$; (c) $\theta = 220°$; (d) $\theta = 315°$.

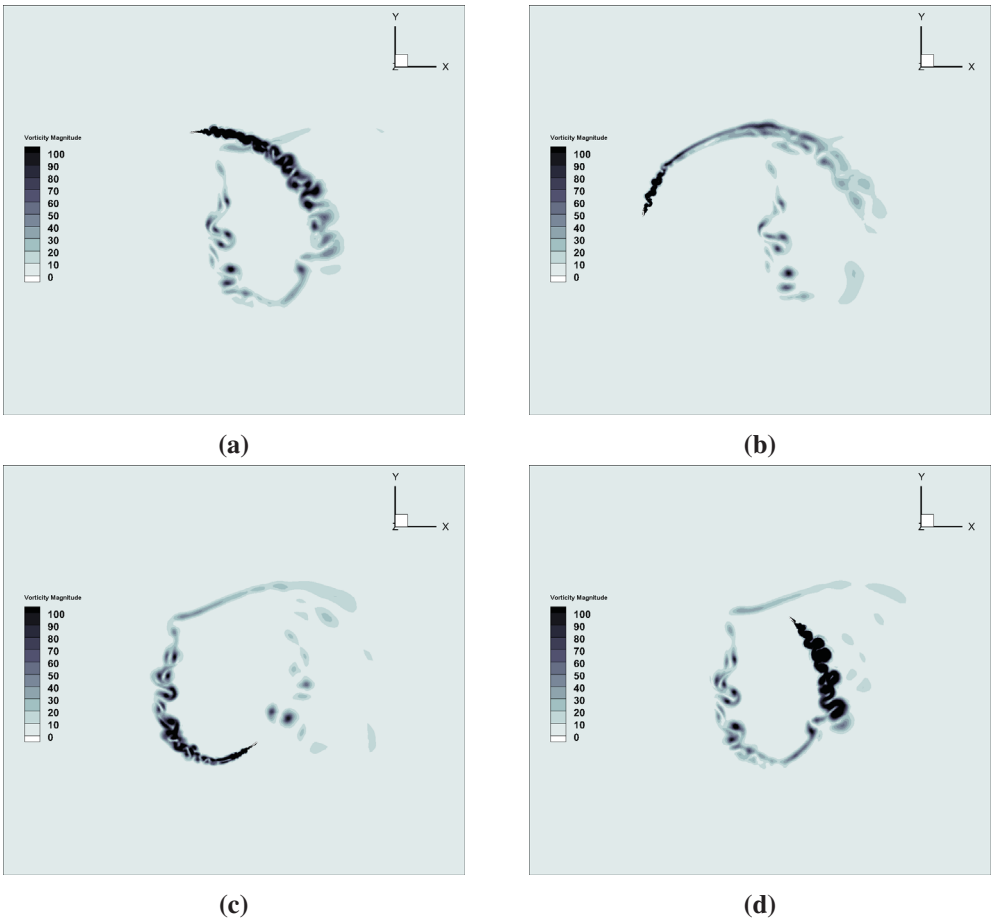


Figure 6.34: The vorticity contour inside the wake [1/s]. (a) $\theta = 5°$; (b) $\theta = 85°$; (c) $\theta = 220°$; (d) $\theta = 315°$.

6.2.4 Comparison of loads on three morphed profiles

In this section, the effect of the airfoil's profiles on power coefficient is studied. A symmetric blade experience the angle of attack up to 15 degrees at this tip speed ratio (see Figure 1.10). The maximum angle may reduces or increases when a chambered airfoil is used. Therefore, plot of AoA versus the azimuthal angle (θ) is shifted up or down based on the deviation angle at the trailing edge. This is seen in Figure 6.35 for the three chosen profiles.

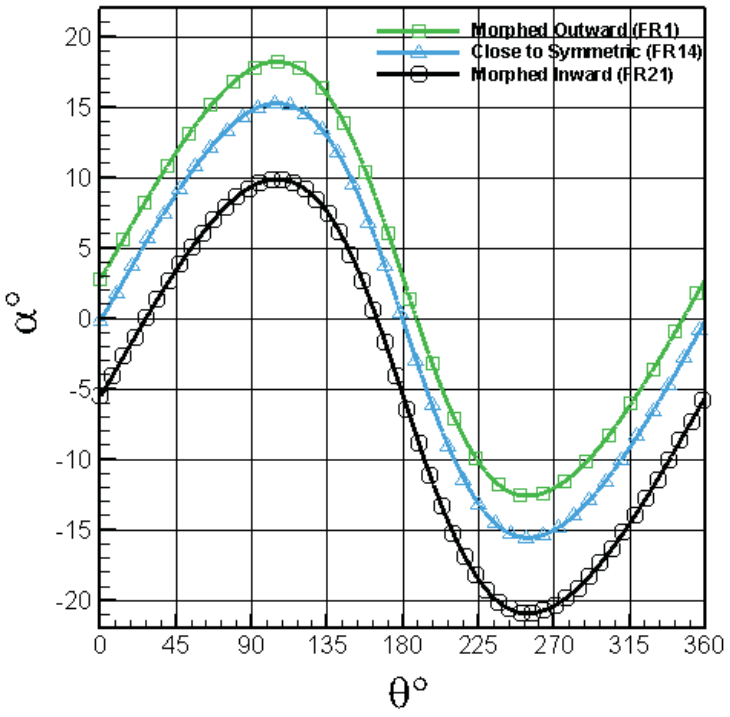


Figure 6.35: Angles of attack versus azimuthal angle for Morphed profiles.

In addition to the angle of attack, the curvature of the blade near the trailing edge are completely different for the three blades. Therefore, it would not be surprising to observe a completely different value of lift and drag for the three morphed blades.

The lift and drag coefficients are plotted in Figure 6.37 and Figure 6.38, respectively. Blade FR14 is very close to the geometry of NACA 0012. Then, lift and drag coefficients are almost zero at the zero azimuthal location. Note that even for a completely symmetric profile, value of lift and drag might not be exactly zero due to hysteresis effect of rotating blades. For the first quarter, the

angle of attack grows for all three cases. However, the morphed outward profile (FR1) sees the highest angle of attack, consequently the lift coefficient keep increasing in the delay of dynamic stall up to azimuthal angle 100 degrees. At this angle of attack, a sharp drop is observed which indicates the existence of a dynamic stall at this point. For a symmetric profile no significant drop is seen due to experience a lower angle of attacks. For the morphed outward profile (FR21) in spite of having a lower AoA there is stall, though weaker, is observed at azimuthal angle 80 degrees. The existence of dynamic stall at a lower AoA is related to the fact that these profiles have different curvatures. Therefore, the stall angle of attack is different for each profile. The drag coefficient sees mostly a positive slope in the first quarter, except for morphed inward (FR21) between 45 to 80 degrees (where stall happens). For the second quarter, when angle of attacks are reduced, then lift and drag also decrease. However, in the case of morphed outward profile, forming and shedding of vortices create some fluctuations at the azimuthal angles between 90 to 135 degrees. An example of these vortices are shown in Figure 6.36.

Figure 6.36: The vorticity contour near the blade for the morphed outward blade (FR1) at $\theta = 120^{\circ}$ [1/s].

During the third quarter of rotation, negative angles of attack are seen for all three profiles. It reaches beyond -20 degrees for the morphed inward (FR21). Both high angle of attack and shape of blade cause to a considerable increase of the drag coefficient for this profile which is almost 3 times higher than the other profiles. It also generate an undesirable lift coefficient in the opposite direction of the blade's path.

Finally, in the last quarter, all three blades recover some of lift, although it is not as high as in the first quarter. Notice that the high fluctuations of lift and drag are related to the continuous

forming and shedding of eddies around the surface profile. These eddies may not be captured with URANS simulations. Therefore, almost all simulations that uses URANS models possibly result in smoother graphs.

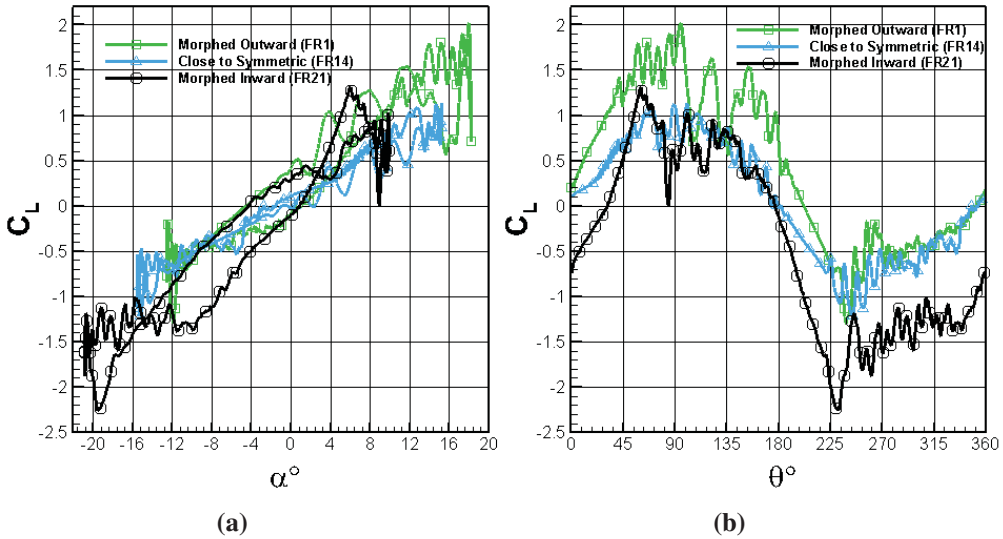
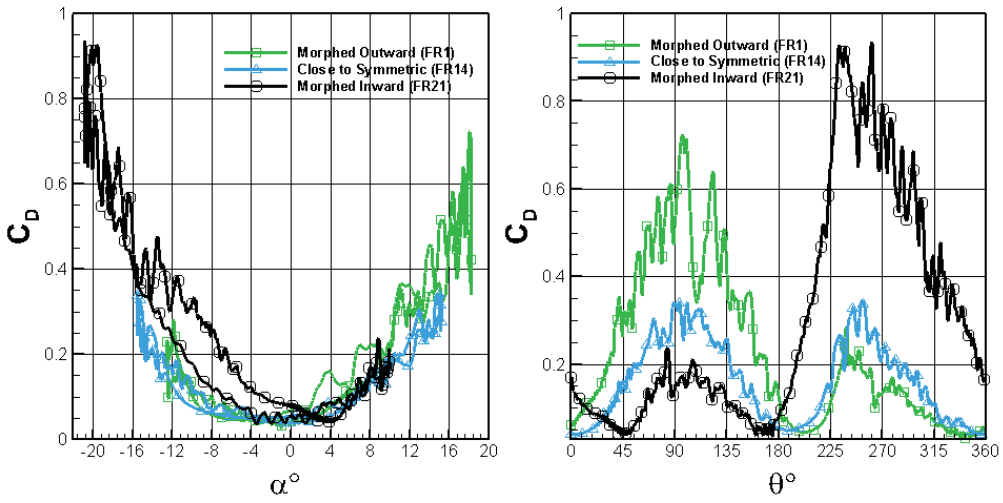


Figure 6.37: Comparison of lift coefficient between three morphed profile.

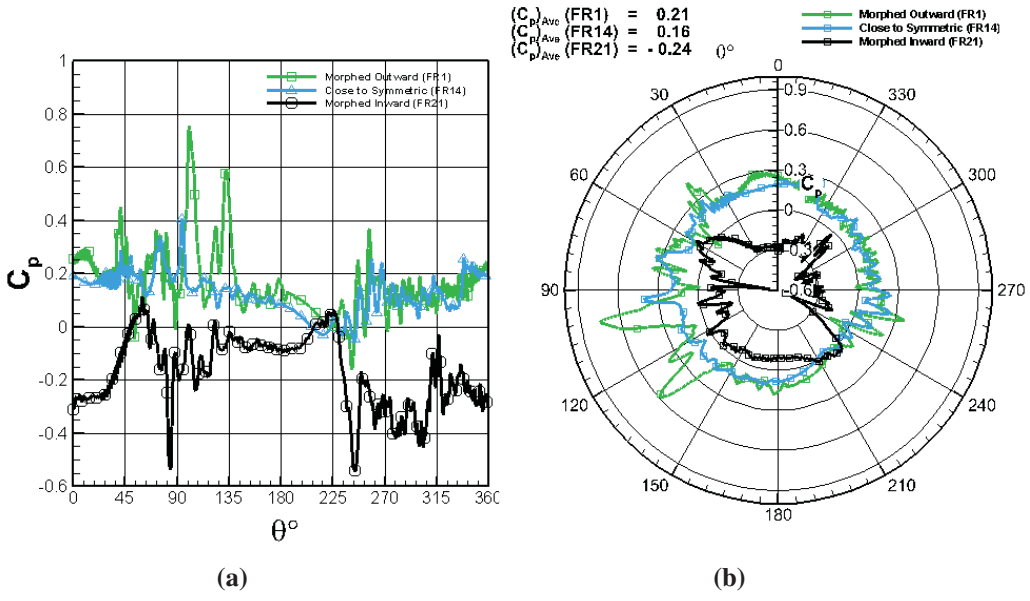
The significance difference profile shape on the power coefficient can be seen in Figure 6.39. Average power changes from -0.24, for the morphed inward profile (FR21), to more than 0.21 for the morphed outward (FR1). This is expected as seen in the lift curve described in the previous section. H-Blade VAWTs are lift-based devices, therefore, extracted power is usually proportional to the lift values. Then, higher value of lift results in higher harnessed energy from the wind.

Note that three selected profile are chosen arbitrary only to demonstrate the performance of the in-house code for large separated flows. Obviously, some profiles may generate power more than FR1, or at lease some portion of the path. Consequently, by morphing the blade at certain points the aerodynamic power can considerably increase. In addition, power coefficient changes with TSR, and current value of TSR does not represent the maximum value of power. A range of TSR should be simulated to identify the maximum C_p.



(a) (b)

Figure 6.38: Comparison of drag coefficient between three morphed profile.



(a) (b)

Figure 6.39: Comparison of power coefficient between three morphed profile.

6.3 Morphing-Blade Vertical Axis Wind Turbine on Mars

In this section, the capability of our code to simulate a morphing blade is studied. As it is seen in the previous section, a morphed-inward profile would not be able to generate any power over one cycle in the average. However, there are some intervals of the azimuthal angle that can generate power. This is also valid for any other profile such as symmetric and morphed-outward blades. According to rapid changes of the angle of attack and the pattern of the flow close to the surface of the blade, a blade at certain positions needs power to be rotated. This is the main reason why the efficiency of VAWT wind turbine is lower than the HAWT ones.

In order to improve the performance of the turbine, the shape of blade can changes dynamically. Therefore, we start from the morphed inward profile FR21 at the angle of 210, where the maximum drag and minimum power is extracted by the blade from the wind. Moving directly from FR21 to the symmetric shape (FR14), or morphed outward profile (FR1) is introduced unphysical numerical error to the solution. Furthermore, one numerical model step time is in the order of $10^{-6} - 10^{-5}$, then it would be an unreal case if the blade profile undergoes such a severe deformation at this interval. Respond time of the flexure box approach is in order of (10^{-1}) at this time [86]. It is feasible to imagine that the interval time can be shorten by one order of magnitude with more sophisticated method. However, reducing that to more than 4 or 5 orders of magnitude would be close to impossible.

Therefore, all 21 Frames are used in order to deform the blade from the morphed inward to symmetric or morphed outward profile. We change the blade from one profile to the next one, while blade sweeps almost 2.5 degrees of its path, and this procedure happens smoothly and gradually. In order to provide more flexibility a new parameter sets the number of sub-frames. Defining this parameter adjusts the rate of deformation.

The same numerical set-up, boundary conditions and grid as the previous section are used for the morphing blade. After five cycles rotating with a fixed blade, the blade goes under deformation and it is deformed the blade at each time step. Thereafter, smoothing the mesh is followed by the entire domain rotation. Herein, we cover ten degrees of this deformation between FR21 to FR17, that covers moving from FR21 to FR17 (Figure 6.40). We use 7000 sub-farms and we use a combination of

the spring and diffusion scheme for the smoothing scheme.

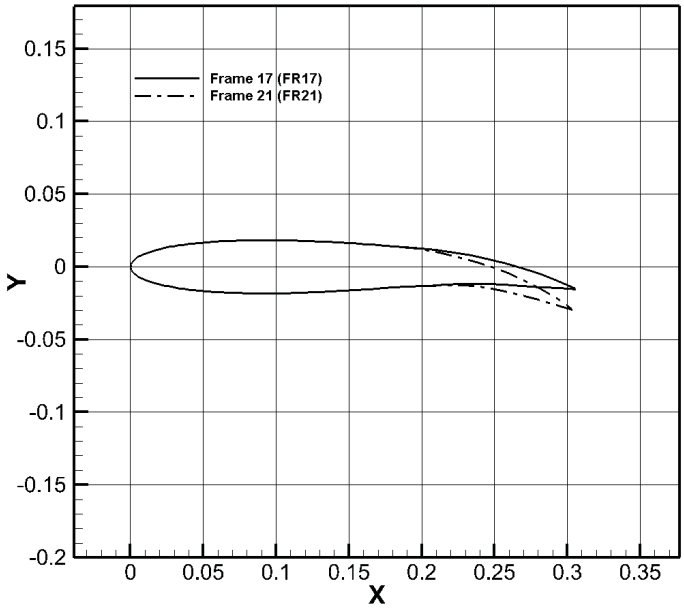
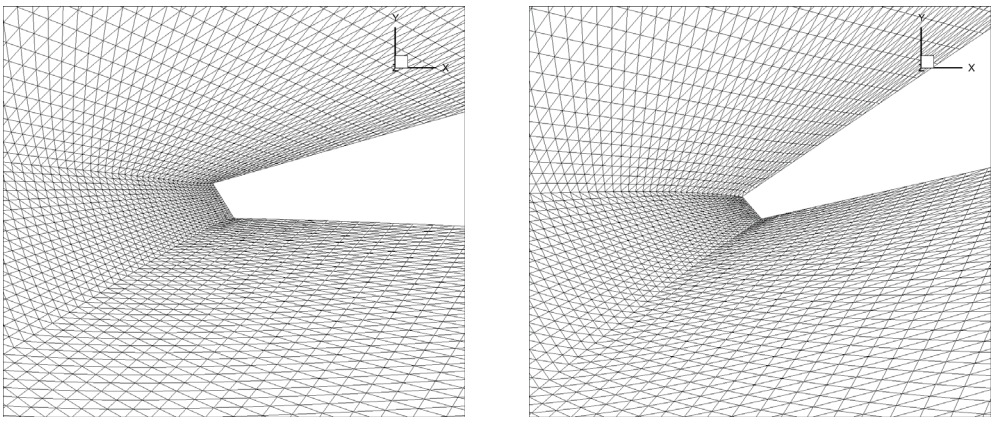


Figure 6.40: Comparison of the profile FR17 and FR21.

Figure 6.41 shows the mesh at the beginning of the deformation (210 degrees) and where it reaches Frame 17 at (220 degrees). The image focus on the trailing edge, because the sharp corners at the trailing edge along with inhomogeneous deformation of each points on the blade is considered as the most challenging part of mesh that needs to be smoothed. It is seen that, even at the trailing edge the skewness and the orthogonality of the mesh is preserved.

The contour of the vorticity are demonstrated between 210 to 220 degrees in Figure 6.42. It is seen that due to the small changes in the shape of the airfoil, almost no discontinuity is observed inside the vorticity contours. Comparison between Figure 6.33c and Figure 6.42d shows that the flow structures does not change over the surface except near the trailing edge. The different shape at the tailing edge, though, changes the shape and pattern of the vortices that shed form the blade. It is expected that these changes have more influence on the flow by continuing the morphing the blade.



(a) **(b)**

Figure 6.41: Mesh deformation from Frame 21 to Frame 17. (a) Frame 21 ($\theta = 210°$); (b) Frame 17 ($\theta = 220°$).

Finally, we look at the power coefficient extracted from the wind in this interval. Power coefficient of Frame 21 and morphing blade are compared. At the beginning of morphing procedure a discontinuity is observed (Figure 6.43), however it rapidly recovers. Notice that this graph uses a different range of axis as in Figure 6.32. While with the same scale, the graph on the top right of Figure 6.43 is obtained. The smoothness of graph shows that both ALE method and the smoothing technique work properly. Full cycle of the blade morphing of the blade will be studied in future work, since the main objective here was to only demonstrate the capability of the developed computational methodology for a morphing blade VAWT.

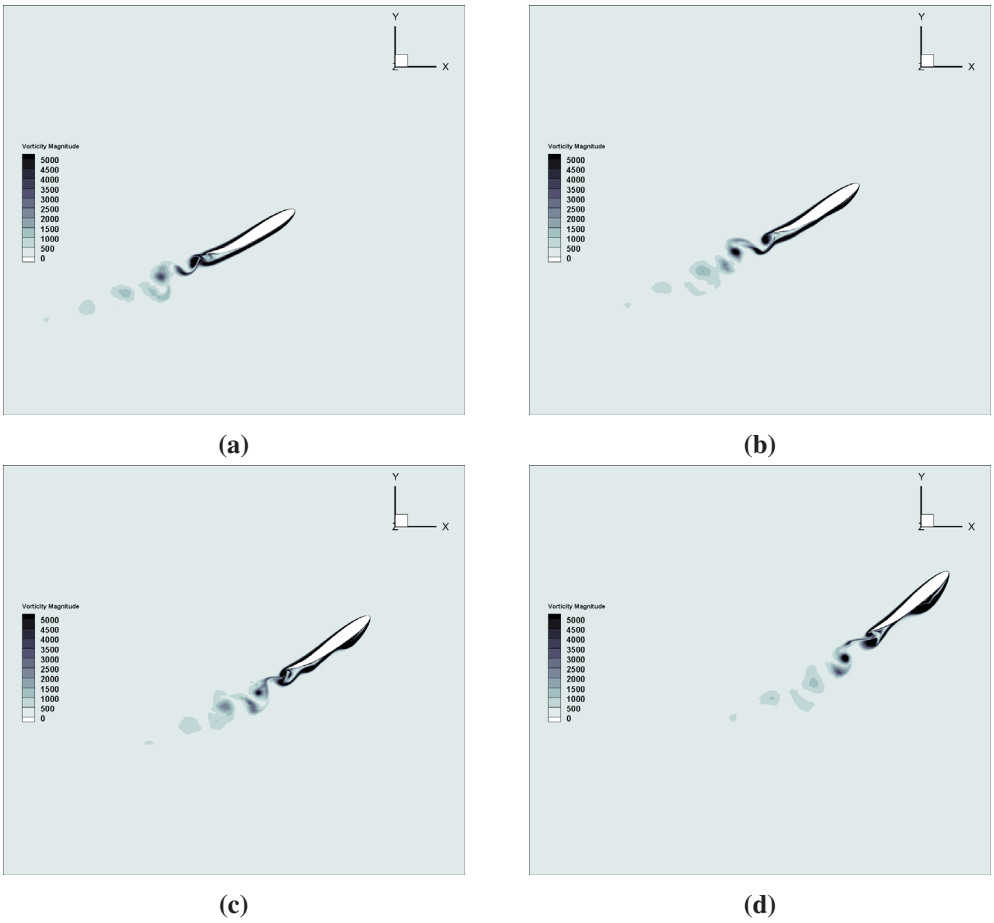


Figure 6.42: The vorticity contour close to Morphing blade's surface [1/s]. (a) $\theta = 210°$; (b) $\theta = 213°$; (c) $\theta = 216°$; (d) $\theta = 220°$.

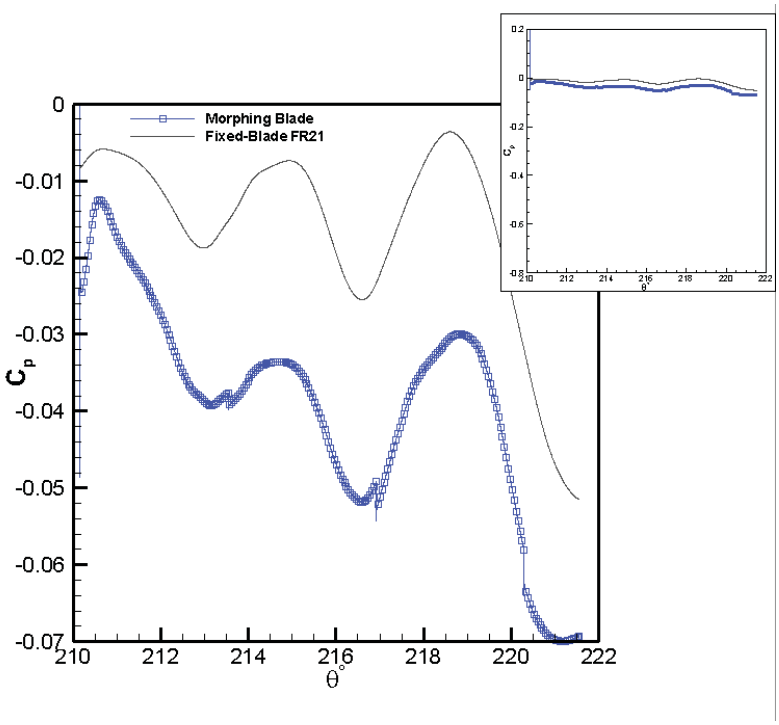


Figure 6.43: Comparison of fixed and morphing blade power generation.

Chapter 7

Conclusions

In this research, an accurate WALE-LES based turbulence model coupled with a scalable parallel in-house code is developed. The code is capable of solving the Navier-Stokes equations up to second order accurate in time and space. It is also capable of tackling a dynamic mesh without loosing its accuracy. The large eddy simulation requires very fine mesh in the near-wall region, consequently using an explicit method limit the time step to an unnecessary small time step, leading to the high computational cost. Therefore, only the implicit solver is considered in the current research.

Chapter 5 presented a convergence study of the code. It included the investigation of the following parameters on the convergence rate on the convergence rate: the CFL number, mesh quality, the Jacobian-matrix approach, and finally comparing the integrated version of GMRES method with the software PETSc.

The results in this chapter showed that the CFL number may be increased toward a very large value while maintaining the convergence of the solution. Although CFL number may increase orders of magnitude on a grid with the AR less than 20, the code can diverge with a low quality grid. This can be an obstacle for the RANS simulations, where, usually the aspect ratio of cells can exceed two or even three orders of magnitude. A grid with the cell aspect ratio less than 100 may also converge, if a smaller CFL number along with the extra number of non-linear iterations are used. Unfortunately, these methods increase the computational cost of the solver. Therefore, it is

suggested that the code is always used with a grid AR that is less than 30. This is not an issue for the large eddy simulation, since to capture the small eddies near walls, the aspect ratio of grid should not exceed 15.

The other important parameter that was investigated in the convergence study chapter was related to number of required non-linear iterations. It has been shown for CFL numbers less than 50, that two outer loops can be sufficient to converge the solution. However, by increasing the CFL number, the number of outer iterations also need to be increased. This conclusion is not general, and the required Newton's iteration may change depending on flow problems and boundary conditions. In any case including inviscid, laminar or turbulent flows, a solver with one Newton's iteration is not suggested for an unsteady problem.

It has been shown that, in spite of the extra programming and complexity of the Jacobian-based model, it converges faster to the solution. In addition, using matrix-free solver increases the number of flops, since the matrix of coefficients needs to be updated at each linear iteration inside the iterative solver.

The flow solver also has been integrated in PETSc. The initial objective was to use a more sophisticated preconditiong methods to speed-up the solution. Then, we tested both approaches (integrated GMRES versus using GMRES inside PETSc) on an Euler problem. The results demonstrated that if one uses the same linear solver and the preconditioning approach on a problem smaller than one million nodes ($(5 \times nodes)$ unknowns), both methods result in very similar convergence rate and CPU times. However, PETSc's efficiency is deteriorated by increasing number of nodes. This can be a subject of investigation in the future.

In the last section of chapter 5 two smoothing methods were compared, the spring and diffusion methods. Both methods work very well when the deformation only includes a homogeneous exasperation or concentration of nodes. In this case the quality of grids including the orthogonality and skewness of the cell are preserved. On the contrary, deformation of a cantilever beam showed that

the spring method is not capable of smoothing the mesh, while the diffusion method smooths the mesh with the minimum changes on the quality of the mesh. It should be noticed that the diffusion method for a large 3-D mesh can be a bottleneck for the solver. Therefore, a combination of both methods was implemented for the VAWT simulation with a morphing blade.

Chapter 6 showed the potential of the WALE model to predict the transition of flow from the laminar to turbulent. The NACA 0012 at two AoAs (9.25 and 12 degrees) were selected for this purpose. The average lift and drag coefficients were comparable with the latest experimental data. All the main coherent turbulent structures can be captured with the in-house solver. Accurate prediction of transition and viscous layer is related to the advantages of the WALE scheme in the vortex dominated regions and better prediction of the turbulent viscosity close to walls. It also highlights the significance of the wiggle detector approach to ease the extra dissipation of the typical upwind schemes. It is worth mentioning here again that the similar problem with the software OpenFOAM would take almost ten times longer with the same number of processors.

VAWT simulations on Mars were showed with some profiles built from the smart material. Results showed that the morphed inward (FR21) and outward (FR1) profiles generate the minimum and maximum power, respectively. In spite of the fact that finding the optimized profile requires simulation of all 21 profiles, the simulations showed the capability of the in-house code to simulate a rotational mesh using the ALE method. In addition, the results showed interesting features of turbulent structures for VAWT simulations that can not be captured with URANS models. These features include: the dynamic stall location, vortex shedding, and transition to the turbulent flow.

Finally, we tested the simulation of the morphing blade VAWT on Mars with the current in-house code. The combination of the spring and diffusion methods helped to preserve the quality of the mesh. In addition, results shows that the dynamic mesh does not add a significant numerical error to the system. However, the power coefficient obtained with a morphing blade requires more simulations in order to have a concrete conclusion about the the power of the morphing blade VAWT on Mars.

In summary, the main contribution of this work is the development, analysis and application of a computer code and methodology for simulating low to moderate Reynolds number flows. Furthermore, a morphing blade in the context of vertical axis wind turbines has been simulated. Now that a powerful LES code is available, in the future, more simulations can be performed to find the best strategy to morph the blade and maximize the power coefficient of all kind of wind turbines that operate at low to moderate Reynolds numbers. The LES code also can be used for accurate simulation of any other moderate Reynolds number application.

Herein are the list of suggestions to improve the code for future simulations.

- Adding capability of the hybrid mesh to the in-house code. It would help the smoothing method in dynamic mesh, and also to improve the convergence for the grid that contains high aspect ratio cells.
- Improving the integration of the in-house code to PETSc. This gives the option to use a more advanced technique for preconditiong the linear system such as Implicit Lower-Upper (ILU) schemes and Algebraic Multigrid (AMG) methods. It can be used to reduce the computational time for long unsteady problems.
- Implement Xeon Phi or GPU in order to run the code on more number of processors and take advantage of the full potential of the code's scalability.
- Comparison of the results of 2D with the in-house 3D LES model, in order to calibrate the RANS models to improve their accuracy for VAWT simulations.

Bibliography

[1] AGGELIKI, K. Vertical windmills - a survey of types and designs [online], 2011. http://www.nasa.gov.

[2] ALAIMO, A., ESPOSITO, A., MESSINEO, A., ORLANDO, C., AND TUMINO, D. 3d cfd analysis of a vertical axis wind turbine. *Energies 8*, 4 (2015), 3013–3033.

[3] AMESTOY, P. R., DUFF, I. S., LEXCELLENT, J.-Y., AND KOSTER, J. Mumps: a general purpose distributed memory sparse solver. In *International Workshop on Applied Parallel Computing* (2000), Springer, pp. 121–130.

[4] BAKER, A. *Finite Element Computational Fluid Mechanics*. Series in Computational Methods in Mechanics and Thermal Sciences. Hemisphere Publishing Company, 1983.

[5] BALAY, S., ABHYANKAR, S., ADAMS, M., BROWN, J., BRUNE, P., BUSCHELMAN, K., EIJKHOUT, V., GROPP, W., KAUSHIK, D., KNEPLEY, M., ET AL. Petsc users manual revision 3.5. *Argonne National Laboratory (ANL)* (2014).

[6] BALDWIN, B. S., AND LOMAX, H. *Thin layer approximation and algebraic model for separated turbulent flows*, vol. 257. American Institute of Aeronautics and Astronautics, 1978.

[7] BATINA, J. T. Unsteady euler airfoil solutions using unstructured dynamic meshes. *AIAA journal 28*, 8 (1990), 1381–1388.

[8] BAUSAS, M. D., AND DANAO, L. A. M. The aerodynamics of a camber-bladed vertical axis wind turbine in unsteady wind. *Energy 93* (2015), 1155–1164.

[9] BENTALEB, Y., SCHALL, E., KOOBUS, B., AND AMARA, M. Low mach investigation of compressible airflow around a generic airship. In *VIII Journées Zaragoza-Pau de Mathématiques Appliquées et de Statistiques: Jaca, Spain, September 15-17, 2003* (2003), Prensas Universitarias de Zaragoza, pp. 509–518.

[10] BERSELLI, L., ILIESCU, T., AND LAYTON, W. J. *Mathematics of large eddy simulation of turbulent flows*. Springer Science & Business Media, 2005.

[11] BETZ, A. *Introduction to the theory of flow machines*. Elsevier, 2014.

[12] BI-CGSTAB, H. V. D. V. a fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems siam j. sci. *Stat. Comput 13* (1992), 631–644.

[13] BLAZEK, J. *Computational Fluid Dynamics: Principles and Applications*. Elsevier Science, 2015.

[14] BLOM, F. J. Considerations on the spring analogy. *International Journal for Numerical Methods in Fluids 32*, 6 (2000), 647–668.

[15] BOTAG. Lift curve of sm701 airfoil showing the relationship between angle of attack and lift coefficient, 2006. https://commons.wikimedia.org/wiki/File:LiftCurve.gif.

[16] BRANDT, J. B., AND SELIG, M. S. Propeller performance data at low reynolds numbers. In *49th AIAA aerospace sciences meeting* (2011), pp. 2011–1255.

[17] CABOT, W., AND MOIN, P. Approximate wall boundary conditions in the large-eddy simulation of high reynolds number flow. *Flow, Turbulence and Combustion 63*, 1-4 (2000), 269–291.

[18] CAMARRI, S., SALVETTI, M. V., KOOBUS, B., AND DERVIEUX, A. A low-diffusion muscl scheme for les on unstructured grids. *Computers & fluids 33*, 9 (2004), 1101–1129.

[19] CHAPMAN, D. R. Computational aerodynamics development and outlook. *AIAA journal 17*, 12 (1979), 1293–1313.

[20] CHEN, C.-C., AND KUO, C.-H. Effects of pitch angle and blade camber on flow characteristics and performance of small-size darrieus vawt. *Journal of Visualization 16*, 1 (2013), 65–74.

[21] CHOI, H., AND MOIN, P. Grid-point requirements for large eddy simulation: Chapmans estimates revisited. *Physics of Fluids (1994-present) 24*, 1 (2012), 011702.

[22] COURANT, R., FRIEDRICHS, K., AND LEWY, H. On the partial difference equations of mathematical physics. *IBM journal 11*, 2 (1967), 215–234.

[23] DAHLSTRÖM, S., AND DAVIDSON, L. Large eddy simulation applied to a high reynolds flow around an airfoil close to stall. In *41st AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January* (2003), pp. 6–9.

[24] DANAO, L. A., QIN, N., AND HOWELL, R. A numerical study of blade thickness and camber effects on vertical axis wind turbines. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy* (2012), 0957650912454403.

[25] DEGAND, C., AND FARHAT, C. A three-dimensional torsional spring analogy method for unstructured dynamic meshes. *Computers & structures 80*, 3 (2002), 305–316.

[26] DEMBO, R. S., EISENSTAT, S. C., AND STEIHAUG, T. Inexact newton methods. *SIAM Journal on Numerical analysis 19*, 2 (1982), 400–408.

[27] DEMIRDŽIĆ, I., LILEK, Ž., AND PERIĆ, M. A collocated finite volume method for predicting flows at all speeds. *International Journal for Numerical Methods in Fluids 16*, 12 (1993), 1029–1050.

[28] DONEA, J., GIULIANI, S., AND HALLEUX, J.-P. An arbitrary lagrangian-eulerian finite element method for transient dynamic fluid-structure interactions. *Computer methods in applied mechanics and engineering 33*, 1-3 (1982), 689–723.

[29] ERICH, H. Wind turbines: fundamentals, technologies, application, economics, 2000.

[30] FALGOUT, R. D., AND YANG, U. M. hypre: A library of high performance preconditioners. In *International Conference on Computational Science* (2002), Springer, pp. 632–641.

[31] FARHAT, C., DEGAND, C., KOOBUS, B., AND LESOINNE, M. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer methods in applied mechanics and engineering 163*, 1 (1998), 231–245.

[32] FERZIGER, J. H., AND PERIC, M. *Computational methods for fluid dynamics*. Springer Science & Business Media, 2012.

[33] FLORES, F., GARREAUD, R., AND MUÑOZ, R. C. Cfd simulations of turbulent buoyant atmospheric flows over complex geometry: Solver development in openfoam. *Computers & Fluids 82* (2013), 1–13.

[34] FRAYSSÉ, V., GIRAUD, L., GRATTON, S., AND LANGOU, J. Algorithm 842: A set of gmres routines for real and complex arithmetics on high performance computers. *ACM Transactions on Mathematical Software (TOMS) 31*, 2 (2005), 228–238.

[35] FUREBY, C., TABOR, G., WELLER, H., AND GOSMAN, A. A comparative study of subgrid scale models in homogeneous isotropic turbulence. *Physics of Fluids (1994-present) 9*, 5 (1997), 1416–1429.

[36] GARNIER, E., ADAMS, N., AND SAGAUT, P. *Large Eddy Simulation for Compressible Flows*. Scientific Computation. Springer Netherlands, 2009.

[37] GERMANO, M., PIOMELLI, U., MOIN, P., AND CABOT, W. H. A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics (1989-1993) 3*, 7 (1991), 1760–1765.

[38] GEURTS, B., SIMAO FERREIRA, C., AND VAN BUSSEL, G. Aerodynamic analysis of a vertical axis wind turbine in a diffuser. In *3rd EWEA Conference-Torque 2010: The Science of making Torque from Wind, Heraklion, Crete, Greece, 28-30 June 2010*, European Wind Energy Association.

[39] GREICIUS, T. Jet propulsion laboratory [online], 2015. http://www.nasa.gov.

[40] HAU, E. Wind turbines: fundamentals, technologies, application, economics. *Springer: Berlin, Germany* (2006).

[41] HESTENES, M. R., AND STIEFEL, E. *Methods of conjugate gradients for solving linear systems*, vol. 49. NBS, 1952.

[42] HIRSCH, C. *Numerical Computation of Internal and External Flows: The Fundamentals of Computational Fluid Dynamics: The Fundamentals of Computational Fluid Dynamics*, vol. 1-2. Butterworth-Heinemann, 2007.

[43] HIRSCH, H., AND MANDAL, A. A cascade theory for the aerodynamic performance of darrieus wind turbines. *Wind Engineering 11*, 3 (1987), 164–175.

[44] HOLME, O. A contribution to the aerodynamic theory of the vertical-axis wind turbine. In *International symposium on wind energy systems* (1977), vol. 1, p. 4.

[45] HOSSAIN, N. Computational mechanics, turbulence modeling [onilne], 2012. https://naimhossain.blogspot.ca/2012/08/turbulence-modeling.html.

[46] ISLAM, M., TING, D. S.-K., AND FARTAJ, A. Aerodynamic models for darrieus-type straight-bladed vertical axis wind turbines. *Renewable and Sustainable Energy Reviews 12*, 4 (2008), 1087–1109.

[47] JASAK, H., AND TUKOVIC, Z. Automatic mesh motion for the unstructured finite volume method. *Transactions of FAMENA 30*, 2 (2006), 1–20.

[48] JIN, X., ZHAO, G., GAO, K., AND JU, W. Darrieus vertical axis wind turbine: Basic research methods. *Renewable and Sustainable Energy Reviews 42* (2015), 212–225.

[49] KANNER, S., AND PERSSON, P.-O. Validation of a high-order large-eddy simulation solver using a vertical-axis wind turbine. *AIAA Journal 54*, 1 (2015), 101–112.

[50] KARYPIS, G., AND KUMAR, V. Metis manual. *University of Minnesota/Department of Science/Army HPC Research Center* (2011).

[51] KLIMENKO, A. Complex competitive systems and competitive thermodynamics. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 371*, 1982 (2013), 20120244.

[52] KNOLL, D. A., AND KEYES, D. E. Jacobian-free newton–krylov methods: a survey of approaches and applications. *Journal of Computational Physics 193*, 2 (2004), 357–397.

[53] KOOBUS, B., AND FARHAT, C. Time-accurate schemes for computing two-and three-dimensional viscous fluxes on unstructured dynamic meshes.

[54] KOOBUS, B., AND FARHAT, C. Second-order time-accurate and geometrically conservative implicit schemes for flow computations on unstructured dynamic meshes. *Computer Methods in Applied Mechanics and Engineering 170*, 1 (1999), 103–129.

[55] KRISHNAN, A. Cfd based design and analysis of building mounted wind turbine with diffuser shaped shroud. July 2015.

[56] LANEVILLE, A., AND VITTECOQ, P. Dynamic stall: the case of the vertical axis wind turbine. *Journal of Solar Energy Engineering 108*, 2 (1986), 140–145.

[57] LAPIN, E. Theoretical performance of vertical axis wind turbines. *American Society of Mechanical Engineers 1* (1975).

[58] LARSEN, H. Summary of a vortex theory for the cyclogiro. In *Proceedings of the 2nd US National conference on Wind Engineering Research.(1975-8), Colorad State University* (1975), vol. 8, p. 1.

[59] LESOINNE, M., AND FARHAT, C. Geometric conservation laws for aeroelastic computations using unstructured dynamic meshes. *AIAA paper*, 95-1709 (1995).

[60] LESOINNE, M., AND FARHAT, C. Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations. *Computer methods in applied mechanics and engineering 134*, 1 (1996), 71–90.

[61] LI, C., ZHU, S., XU, Y.-L., AND XIAO, Y. 2.5 d large eddy simulation of vertical axis wind turbine in consideration of high angle of attack flow. *Renewable energy 51* (2013), 317–330.

[62] LUO, H., BAUM, J. D., AND LÖHNER, R. A fast, matrix-free implicit method for compressible flows on unstructured grids. In *Sixteenth international conference on numerical methods in fluid dynamics* (1998), Springer, pp. 73–78.

[63] LUO, H., BAUM, J. D., AND LÖHNER, R. An accurate, fast, matrix-free implicit method for computing unsteady flows on unstructured grids. *Computers & fluids 30*, 2 (2001), 137–159.

[64] MAGRI, L., AND GALVANETTO, U. Example of a non-smooth hopf bifurcation in an aeroelastic system. *Mechanics Research Communications 40* (2012), 26–33.

[65] MARTIN, M. P., PIOMELLI, U., AND CANDLER, G. V. Subgrid-scale models for compressible large-eddy simulations. *Theoretical and Computational Fluid Dynamics 13*, 5 (2000), 361–376.

[66] MCCROSKEY, W. J. The phenomenon of dynamic stall. Tech. rep., DTIC Document, 1981.

[67] MCCROSKEY, W. J., MCALISTER, K., CARR, L., PUCCI, S., LAMBERT, O., AND INDERGRAND, R. Dynamic stall on advanced airfoil sections. *Journal of the American Helicopter Society 26*, 3 (1981), 40–50.

[68] MENEVEAU, C., LUND, T. S., AND CABOT, W. H. A lagrangian dynamic subgrid-scale model of turbulence. *Journal of Fluid Mechanics 319* (1996), 353–385.

[69] MENTER, F. R. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA journal 32*, 8 (1994), 1598–1605.

[70] MIAU, J., LIANG, S., YU, R., HU, C., LEU, T., CHENG, J., AND CHEN, S. Design and test of a vertical-axis wind turbine with pitch control. In *Applied Mechanics and Materials* (2012), vol. 225, Trans Tech Publ, pp. 338–343.

[71] MOHAMED, M., ALI, A., AND HAFIZ, A. Cfd analysis for h-rotor darrieus turbine as a low speed wind energy converter. *Engineering Science and Technology, an International Journal 18*, 1 (2015), 1–13.

[72] MOIN, P., AND KIM, J. Numerical investigation of turbulent channel flow. *Journal of fluid mechanics 118* (1982), 341–377.

[73] MUNSON, B., ROTHMAYER, A., AND OKIISHI, T. *Fundamentals of Fluid Mechanics, 7th Edition*. John Wiley & Sons, Incorporated, 2012.

[74] MURACA, R., STEPHENS, M. V., AND DAGENHART, J. R. Theoretical performance of cross-wind axis turbines with results for a catenary vertical axis configuration.

[75] NELSON, J. Nasa, jet propulsion laboratory [online]. http://mars.nasa.gov.

[76] NELSON, J. Nasa, jet propulsion laboratory [online], 2015. http://jpl.nasa.gov.

[77] NELSON, J. National aeronautics and space administration [online], 2015. http://mars.jpl.nasa.gov.

[78] NICOUD, F., BAGGETT, J., MOIN, P., AND CABOT, W. Large eddy simulation wall-modeling based on suboptimal control theory and linear stochastic estimation. *Physics of Fluids (1994-present) 13*, 10 (2001), 2968–2984.

[79] NICOUD, F., AND DUCROS, F. Subgrid-scale stress modelling based on the square of the velocity gradient tensor. *Flow, turbulence and Combustion 62*, 3 (1999), 183–200.

[80] NIKITIN, N., NICOUD, F., WASISTHO, B., SQUIRES, K., AND SPALART, P. An approach to wall modeling in large-eddy simulations. *Physics of Fluids (1994-present) 12*, 7 (2000), 1629–1632.

[81] NKONGA, B., AND GUILLARD, H. Godunov type method on non-structured meshes for three-dimensional moving boundary problems. *Computer methods in applied mechanics and engineering 113*, 1 (1994), 183–204.

[82] OHYA, Y., AND KARASUDANI, T. A shrouded wind turbine generating high output power with wind-lens technology. *Energies 3*, 4 (2010), 634–649.

[83] OSUSKY, M., HICKEN, J. E., AND ZINGG, D. W. A parallel newton-krylov-schur flow solver for the navier-stokes equations using the sbp-sat approach. In *48th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, Florida, AIAA–2010–116* (2010).

[84] OWEN, T. The composition and early history of the atmosphere of mars. *Mars 1* (1992), 818–834.

[85] PANKONIEN, A., AND INMAN, D. J. Experimental testing of spanwise morphing trailing edge concept. In *SPIE Smart Structures and Materials+ Nondestructive Evaluation and Health Monitoring* (2013), International Society for Optics and Photonics, pp. 868815–868815.

[86] PANKONIEN, A. M. *SMART MATERIAL WING MORPHING FOR UNMANNED AERIAL VEHICLES*. PhD thesis, University of Michigan, 2015.

[87] PARASCHIVOIU, I. Double-multiple streamtube model for darrieus in turbines. In *Wind turbine dynamics* (1981).

[88] PARASCHIVOIU, M., KOMEILI, M., AND ZADEH, S. N. Mesh convergence study for 2-d straight-blade vertical axis wind turbine simulations and estimation for 3-d simulations. *Transactions of the Canadian Society for Mechanical Engineering, 38*, 4 (2014).

[89] PIOMELLI, U. Large eddy simulations in 2030 and beyond. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 372*, 2022 (2014), 20130320.

[90] PIOMELLI, U., FERZIGER, J., MOIN, P., AND KIM, J. New approximate boundary conditions for large eddy simulations of wall-bounded flows. *Physics of Fluids A: Fluid Dynamics (1989-1993) 1*, 6 (1989), 1061–1068.

[91] POPE, S. Turbulent flows, 771 pp, 2000.

[92] REID, J. K. On the method of conjugate gradients for the solution of large sparse systems of linear equations, 1971.

[93] ROE, P. L. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of computational physics 43*, 2 (1981), 357–372.

[94] SAAD, Y., AND SCHULTZ, M. H. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing 7*, 3 (1986), 856–869.

[95] SATHAYE, S. S. *Lift distributions on low aspect ratio wings at low Reynolds numbers*. PhD thesis, Worcester polytechnic institute, 2004.

[96] SCHLICHTING, H. Boundary-layer theory.

[97] SCOTTI, A., MENEVEAU, C., AND FATICA, M. Dynamic smagorinsky model on anisotropic grids. *Physics of Fluids (1994-present) 9*, 6 (1997), 1856–1858.

[98] SINGH, R. K., AHMED, M. R., ZULLAH, M. A., AND LEE, Y.-H. Design of a low reynolds number airfoil for small horizontal axis wind turbines. *Renewable energy 42* (2012), 66–76.

[99] SMAGORINSKY, J. General circulation experiments with the primitive equations: I. the basic experiment*. *Monthly weather review 91*, 3 (1963), 99–164.

[100] SMITH, A., AND CEBECI, T. Numerical solution of the turbulent-boundary-layer equations. Tech. rep., DTIC Document, 1967.

[101] SMITH, K. San fransico alerion express fleet [online], 2013. https://sfbayalerion.com/tech-corner/.

[102] SPALART, P. R. Detached-eddy simulation. *Annual review of fluid mechanics 41* (2009), 181–202.

[103] SPALART, P. R., AND ALLMARAS, S. R. A one equation turbulence model for aerodinamic flows. *AIAA journal 94* (1992).

[104] STOKES, G. G. *On the effect of the internal friction of fluids on the motion of pendulums*, vol. 9. Pitt Press, 1851.

[105] STRELETS, M. Detached eddy simulation of massively separated flows. In *AIAA, Aerospace Sciences Meeting and Exhibit, 39 th, Reno, NV* (2001).

[106] TAJALLIPOUR, N., BABAEE OWLAM, B., AND PARASCHIVOIU, M. Self-adaptive upwinding for large eddy simulation of turbulent flows on unstructured elements. *Journal of Aircraft 46*, 3 (2009), 915–926.

[107] TAN, J., KOMEILI, M., PANKONIEN, A., INMAN, D., AND PARASCHIVOIU, M. Smart morphing blade for vertical axis wind turbines. In 24^{th} *International Congress of Theoretical and Applied Mechanics, Montreal* (2016).

[108] TCHAKOUA, P., WAMKEUE, R., OUHROUCHE, M., TAMEGHE, T. A., AND EKEMB, G. A new approach for modeling darrieus-type vertical axis wind turbine rotors using electrical equivalent circuit analogy: Basis of theoretical formulations and model development. *Energies 8*, 10 (2015), 10684–10717.

[109] TEMPLIN, R. Aerodynamic performance theory for the nrc vertical-axis wind turbine. Tech. rep., National Aeronautical Establishment, Ottawa, Ontario (Canada), 1974.

[110] TUREK, S. *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approache*, vol. 6. Springer Science & Business Media, 1999.

[111] TURKEL, E. Preconditioned methods for solving the incompressible and low speed compressible equations. *Journal of computational physics 72*, 2 (1987), 277–298.

[112] VAN BUSSEL, G., MERTENS, S., POLINDER, H., AND SIDLER, H. Turby®: concept and realisation of a small vawt for the built environment. In *EAWE/EWEA conference, Delft* (2004).

[113] VAN DRIEST, E. R. On turbulent flow near a wall. *Journal of the Aeronautical Sciences* (2012).

[114] VERSTEEG, H., AND MALALASEKERA, W. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Pearson Education Limited, 2007.

[115] VIOZAT, C. *Implicit upwind schemes for low Mach number compressible flows*. PhD thesis, Inria, 1997.

[116] WATANABE, K., TAKAHASHI, S., AND OHYA, Y. Application of a diffuser structure to vertical-axis wind turbines. *Energies 9*, 6 (2016), 406.

[117] WAYNE. Alternative energy tutorials, wind energy turbines [online], 2016. http://www.alternative-energy-tutorials.com/wind-energy/wind-energy.html/.

[118] WEISS, K., IURICICH, F., FELLEGARA, R., AND DE FLORIANI, L. A primal/dual representation for discrete morse complexes on tetrahedral meshes. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 361–370.

[119] WILCOX, D. C., ET AL. *Turbulence modeling for CFD*, vol. 2. DCW industries La Canada, CA, 1998.

[120] WILSON, R. E., AND LISSAMAN, P. B. Applied aerodynamics of wind power machines. Tech. rep., Oregon State Univ., Corvallis (USA), 1974.

[121] WOLFF, T., ERNST, B., AND SEUME, J. Aerodynamic behavior of an airfoil with morphing trailing edge for wind turbine applications. In *Journal of Physics: Conference Series* (2014), vol. 524, IOP Publishing, p. 012018.

[122] WONG, P., AND ZINGG, D. W. Three-dimensional aerodynamic computations on unstructured grids using a newton–krylov approach. *Computers & Fluids 37*, 2 (2008), 107–120.

[123] XIAO, Q., LIU, W., AND INCECIK, A. Flow control for vatt by fixed and oscillating flap. *Renewable energy 51* (2013), 141–152.

Appendix A

Jacobian Matrices

Jacobian Matrix of the convective flux

The conservative form of the Euler Equation can be written in the following form,

$$\frac{\partial W}{\partial t} + {}^{c}A_1 \frac{\partial W}{\partial x_1} + {}^{c}A_2 \frac{\partial W}{\partial x_2} + {}^{c}A_3 \frac{\partial W}{\partial x_3} = 0 \tag{A.1}$$

$W = \{\rho, \rho u_1, \rho u_2, \rho u_3, \rho e\}$ are the conservative variables. ${}^{c}A_1$, ${}^{c}A_2$, ${}^{c}A_3$ are the convective flux Jacobian and they are given,

$${}^{c}A_1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ \frac{\gamma-1}{2}V^2 - u_1^2 & (3-\gamma)u_1 & -(\gamma-1)u_2 & -(\gamma-1)u_3 & \gamma-1 \\ -u_1u_2 & u_2 & u_1 & 0 & 0 \\ -u_1u_3 & u_3 & 0 & u_1 & 0 \\ ((\gamma-1)V^2 - h)u_1 & h-(\gamma-1)u_1^2 & -(\gamma-1)u_1u_2 & -(\gamma-1)u_1u_3 & \gamma u_1 \end{pmatrix} \tag{A.2}$$

$$
{}^{c}A_2 = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 \\
-u_1u_2 & u_2 & u_1 & 0 & 0 \\
\frac{\gamma-1}{2}V^2 - u_2^2 & -(\gamma-1)u_1 & (3-\gamma)u_2 & -(\gamma-1)u_3 & \gamma-1 \\
-u_2u_3 & 0 & u_3 & u_2 & 0 \\
((\gamma-1)V^2 - h)u_2 & -(\gamma-1)u_1u_2 & h-(\gamma-1)u_2^2 & -(\gamma-1)u_2u_3 & \gamma u_2
\end{pmatrix} \tag{A.3}
$$

$$
{}^{c}A_3 = \begin{pmatrix}
0 & 0 & 0 & 1 & 0 \\
-u_1u_3 & u_3 & 0 & u_1 & 0 \\
-u_2u_3 & 0 & u_3 & u_2 & 0 \\
\frac{\gamma-1}{2}V^2 - u_3^2 & -(\gamma-1)u_1 & -(\gamma-1)u_2 & (3-\gamma)u_3 & \gamma-1 \\
((\gamma-1)V^2 - h)u_3 & -(\gamma-1)u_1u_3 & -(\gamma-1)u_2u_3 & h-(\gamma-1)u_3^2 & \gamma u_3
\end{pmatrix} \tag{A.4}
$$

Then matrix A is given by,

$$
A = {}^{c}\, A_1.n_1 + {}^{c}\, A_2.n_2 + {}^{c}\, A_3.n_3 \tag{A.5}
$$

In the matrix form it is given by,

$$
{}^{c}A = \begin{pmatrix}
0 & n_1 & n_2 & n_3 & 0 \\
\frac{\gamma-1}{2}V^2n_1 - u_1(\vec{V}.\vec{n}) & (2-\gamma)u_1n_1 + \vec{V}.\vec{n} & -(\gamma-1)u_2n_1 + u_1n_1 & -(\gamma-1)u_3n_1 + u_1n_2 & (\gamma-1)n_1 \\
\frac{\gamma-1}{2}V^2n_2 - u_2(\vec{V}.\vec{n}) & -(\gamma-1)u_1n_2 + u_2n_1 & (2-\gamma)u_2n_2 + \vec{V}.\vec{n} & -(\gamma-1)u_3n_2 + u_2n_3 & (\gamma-1)n_2 \\
\frac{\gamma-1}{2}V^2n_3 - u_3(\vec{V}.\vec{n}) & -(\gamma-1)u_1n_3 + u_3n_1 & -(\gamma-1)u_2n_3 + u_3n_2 & (2-\gamma)u_3n_3 + \vec{V}.\vec{n} & (\gamma-1)n_3 \\
(\frac{\gamma-1}{2}V^2 - h)(\vec{V}.\vec{n}) & (hn_1 - (\gamma-1)u_1)(\vec{V}.\vec{n}) & (hn_2 - (\gamma-1)u_2)(\vec{V}.\vec{n}) & (hn_3 - (\gamma-1)u_3)(\vec{V}.\vec{n}) & \gamma(\vec{V}.\vec{n})
\end{pmatrix} \tag{A.6}
$$

Jacobian Matrix of Arbitrary Lagrangian-Eulerian Method(ALE)

The convertive fluxex in a moving mesh with velocity of $\dot{x}$ is modified to,

$$ {}^{c}F_{ALE}(W) = {}^{c}F(W) - \dot{x}W \tag{A.7} $$

Then ALE equation is given by,

$$ \frac{\partial W}{\partial t} + {}^{c}A_1 \frac{\partial W}{\partial x_1} + {}^{c}A_2 \frac{\partial W}{\partial x_2} + {}^{c}A_3 \frac{\partial W}{\partial x_3} - \dot{x}\nabla.W = 0 \tag{A.8} $$

Then the Jacobian is given by,

$$ {}^{c}A = \begin{pmatrix} -\dot{x} & n_1 & n_2 & n_3 & 0 \\ \frac{\gamma-1}{2}V^2 n_1 - u_1(\vec{V}.\vec{n}) & (2-\gamma)u_1 n_1 + \vec{V}.\vec{n} - \dot{x} & -(\gamma-1)u_2 n_1 + u_1 n_1 & -(\gamma-1)u_3 n_1 + u_1 n_2 & (\gamma-1)n_1 \\ \frac{\gamma-1}{2}V^2 n_2 - u_2(\vec{V}.\vec{n}) & -(\gamma-1)u_1 n_2 + u_2 n_1 & (2-\gamma)u_2 n_2 + \vec{V}.\vec{n} - \dot{x} & -(\gamma-1)u_3 n_2 + u_2 n_3 & (\gamma-1)n_2 \\ \frac{\gamma-1}{2}V^2 n_3 - u_3(\vec{V}.\vec{n}) & -(\gamma-1)u_1 n_3 + u_3 n_1 & -(\gamma-1)u_2 n_3 + u_3 n_2 & (2-\gamma)u_3 n_3 + \vec{V}.\vec{n} - \dot{x} & (\gamma-1)n_3 \\ (\frac{\gamma-1}{2}V^2 - h)(\vec{V}.\vec{n}) & (hn_1 - (\gamma-1)u_1)(\vec{V}.\vec{n}) & (hn_2 - (\gamma-1)u_2)(\vec{V}.\vec{n}) & (hn_3 - (\gamma-1)u_3)(\vec{V}.\vec{n}) & \gamma(\vec{V}.\vec{n}) - \dot{x} \end{pmatrix} \tag{A.9} $$

Appendix B

PETSc Interface

```
c
c This subroutine call PETSc to solve linear system Ax=B
c

SUBROUTINE CALLPETSC

c ------------------------------------------------
c In_house code header files
c ------------------------------------------------

INCLUDE "Param3D.h"
INCLUDE "Paral3D.h"
INCLUDE "ParaPETSC.h"

c ------------------------------------------------
c PETSc header files
c ------------------------------------------------

#include <finclude/petscsys.h>
```

```
#include <finclude/petscvec.h>
#include <finclude/petscmat.h>
#include <finclude/petscksp.h>
#include <finclude/petscpc.h>

c ----------------------------------------------
c   Define local variables
c ----------------------------------------------

INTEGER    bs
PARAMETER (bs=5)
REAL*8       SolGlob(bs*nsmaxg)
REAL*8       SolGlobTemp(bs*nsmaxg)
REAL*8       SolTemp(bs*nsmaxg)
REAL*8       tt1,tt2

INTEGER      l,ll
INTEGER      buffer
INTEGER      nsarray(mxghd),istartary(mxghd)
INTEGER      istart
INTEGER      ig,is
INTEGER      nsmSum
INTEGER      myunit2,lensd,kksave
CHARACTER*80 based
CHARACTER*80 hyprePcName
CHARACTER*80 solverType

CHARACTER*80 name
INTEGER myunit
```

```
c ----------------------------------------------
c Define PETSC variables
c ----------------------------------------------

PetscErrorCode ierr

PetscInt     n,NN,low,high
PetscInt     iglobal,i,ione,j
PetscInt     first,stride
PetscInt     its
PetscInt     ii,jj
PetscInt     ix(bs),ix2(bs),nsglob
PetscInt     itwo
PetscInt     idxn(bs),idxm(bs)
PetscInt     nMat,mMat
PetscInt     maxits
PetscInt     tempapp(nsmaxg)
PetscInt     temppetsc(nsmaxg)
PetscInt     mapToPetsc(nsmaxg)
PetscInt     mapToApp(nsmaxg)
PetscInt     myapp(nsmaxg)
PetscInt     mypetsc(nsmaxg)
PetscInt     ksp_gmres_restart
PetscInt     ksp_lgmres_augment

PetscScalar  value, zero
PetscScalar  vVal(bs)
PetscScalar  yy(bs),zz(bs)
```

```
PetscScalar  mVal(bs,bs)

PetscReal    rtol,abstol

PetscOffset  i_x

Vec          x, y, z, bvec
VecScatter   ctx
KSP          ksp
Mat          Amat
PC           pc

save         Amat
save         mapToPetsc
save         mapToApp

hyprePCName  = "boomeramg"

c ------------------------------------------------
c    bs     - block size
c ------------------------------------------------

ione         = 1
itwo         = 2
zero         = 0.0d0
ierr         = 0

c ------------------------------------------------
c Set Tolerance Parameters
```

```
c ---------------------------------------------

solverType  = "KSPGMRES"
rtol        = 1.e-8
abstol      = 1.e-6
maxits      = 200

c ---------------------------------------------
c Initialization
c ---------------------------------------------

Do i=1,nsm*bs

SolTemp(i) = 0.d0

END Do



c ---------------------------------------------
c  Set up Right hand vector and solution vertor
c ---------------------------------------------

CALL VecCreateMPI(PETSC_COMM_WORLD,nsm*bs,PETSC_DETERMINE
&                  ,bvec,ierr)

C
C     Create solution solution vector
C
```

```
CALL VecDuplicate(bvec,x,ierr)

c
c     Initialize solution to zero
c

CALL VecSet(x,zero,ierr)

c -----------------------------------------------
c  Store right had side values in 1*5 block
c -----------------------------------------------

DO  i=1,nsm
is             = ver4Code(i)

ii             = mapToPETSC(igrefdT(is))
ix(1)          = (ii-1)*bs

ix(2)          = ix(1) + 1
ix(3)          = ix(2) + 1
ix(4)          = ix(3) + 1
ix(5)          = ix(4) + 1

vVal(1)        = vec_x(1,i)
vVal(2)        = vec_x(2,i)
vVal(3)        = vec_x(3,i)
vVal(4)        = vec_x(4,i)
vVal(5)        = vec_x(5,i)
CALL VecSetValues(bvec,bs,ix,vVal
```

```
&                        ,INSERT_VALUES,ierr)
END DO
c ----------------------------------------------
c      Assemble Vector for MPI
c ----------------------------------------------

CALL VecAssemblyBegin(bvec,ierr)
CALL VecAssemblyEnd(bvec,ierr)
c
c     create Matrix entries
c
CALL MatCreate(PETSC_COMM_WORLD, Amat, ierr)
c
c Define Matrix type
c
CALL MatSetType(AMat, MATMPIBAIJ, ierr)
c
c Set Matrix Global Size
c

CALL MatSetSizes(Amat, nsm*bs, nsm*bs, PETSC_DETERMINE
&                ,PETSC_DETERMINE, ierr)

c
c preallocate Matrix
c
CALL MatSetUp(Amat,ierr)
c  ----------------------------------------------
c  store Matrix values in a 5*5 block format
```

```
c  ------------------------------------------------
DO i=1,nsm

is               = ver4Code(i)



IF (is.eq.0) Then
print*,"There is an error in ver4Code "
&              ,"in Partition: ", ipd, "node: ", i
Call TILT
END IF

c ------------------------------------------------
c Global row Index
c ------------------------------------------------

c         if (igrefd(is).EQ.0) Then
c             print*, "ipd: ", ipd, "is: ", is, "i : ", i
c             Call TILT
c        end if
ii               = mapToPETSC(igrefdT(is))

idxm(1)          = (ii-1)
idxm(2)          = idxm(1) + 1
idxm(3)          = idxm(2) + 1
idxm(4)          = idxm(3) + 1
idxm(5)          = idxm(4) + 1

DO j=1,nsn(i)
```

```
c ------------------------------------------------
c Global Column Index
c ------------------------------------------------

jj           = mapToPETSC(colIndx(i,j))

idxn(1)      = (jj-1)
idxn(2)      = idxn(1) + 1
idxn(3)      = idxn(2) + 1
idxn(4)      = idxn(3) + 1
idxn(5)      = idxn(4) + 1

mVal(1,1)    = values(1,1,i,j)
mVal(2,1)    = values(1,2,i,j)
mVal(3,1)    = values(1,3,i,j)
mVal(4,1)    = values(1,4,i,j)
mVal(5,1)    = values(1,5,i,j)

mVal(1,2)    = values(2,1,i,j)
mVal(2,2)    = values(2,2,i,j)
mVal(3,2)    = values(2,3,i,j)
mVal(4,2)    = values(2,4,i,j)
mVal(5,2)    = values(2,5,i,j)

mVal(1,3)    = values(3,1,i,j)
mVal(2,3)    = values(3,2,i,j)
mVal(3,3)    = values(3,3,i,j)
mVal(4,3)    = values(3,4,i,j)
```

```
mVal(5,3)    = values(3,5,i,j)

mVal(1,4)    = values(4,1,i,j)
mVal(2,4)    = values(4,2,i,j)
mVal(3,4)    = values(4,3,i,j)
mVal(4,4)    = values(4,4,i,j)
mVal(5,4)    = values(4,5,i,j)

mVal(1,5)    = values(5,1,i,j)
mVal(2,5)    = values(5,2,i,j)
mVal(3,5)    = values(5,3,i,j)
mVal(4,5)    = values(5,4,i,j)
mVal(5,5)    = values(5,5,i,j)
CALL MatSetValuesBlocked(Amat,1,idxm(1),1,idxn(1),mVal,
&                        INSERT_VALUES, ierr)
End DO
End Do
CALL MatAssemblyBegin(Amat, MAT_FINAL_ASSEMBLY, ierr)
CALL MatAssemblyEnd(Amat, MAT_FINAL_ASSEMBLY, ierr)
End If
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
c         Create the linear solver and set various options
c - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

c  Create linear solver context
c ---------------------------------------------

CALL KSPCreate(PETSC_COMM_WORLD, ksp, ierr)
```

```
c -----------------------------------------------
c  Set operators. Here the matrix that defines the linear system
c  also serves as the preconditioning matrix.
c -----------------------------------------------

CALL KSPSetOperators(ksp, Amat, Amat, ierr)
c -----------------------------------------------
c set-up solver type
c -----------------------------------------------

CALL KSPGMRESSetRestart(ksp, ksp_gmres_restart, ierr)
c
c Set-up preconditionar
c
CALL KSPGetPC(ksp, pc, ierr)
CALL  KSPSetTolerances(ksp,rtol,PETSC_DEFAULT_REAL
&     ,PETSC_DEFAULT_REAL, maxits, ierr)



c - - - - - - - - - - - - - - - - - - - - - - - - -
c         Solve the linear system
c - - - - - - - - - - - - - - - - - - - - - - - - -

CALL KSPSolve(ksp,bvec,x,ierr)

c -----------------------------------------------
c Retrieve the values from PETSC solver
c -----------------------------------------------
```

```
Do is=1,ns
dx(1,is) = 0.d0
dx(2,is) = 0.d0
dx(3,is) = 0.d0
dx(4,is) = 0.d0
dx(5,is) = 0.d0
End Do



Do i=1,nsm


ix(1)            = (i-1)*bs + istart
ix(2)            = ix(1) + 1
ix(3)            = ix(2) + 1
ix(4)            = ix(3) + 1
ix(5)            = ix(4) + 1
CALL  VecGetValues(x, bs, ix, yy, ierr)
is               = ver4Code(i)
dx(1,is)         = yy(1)
dx(2,is)         = yy(2)
dx(3,is)         = yy(3)
dx(4,is)         = yy(4)
dx(5,is)         = yy(5)


END Do


c - - - - - - - - - - - - - - - - - - - - - - - - - -
c           Destroy the solver
c - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
CALL VecDestroy(x, ierr)

CALL VecDestroy(bvec, ierr)

If ( inewt .EQ. nbnewton) Then
CALL MatDestroy(Amat, ierr)
End If

CALL KSPDestroy(ksp, ierr)
RETURN
End SUBROUTINE
```

Appendix C

Fourier's Coefficients for Airfoils' Profile

The Fourier's expansion to extract points on the blade is given by,

$$y(x) = a_0 + \sum_{n=1}^{6}(a_n \cos(nfx) + b_n \sin(nfx)), \tag{C.1}$$

and the coefficients are listed in the following table (units are in millimetre),

Table C.1: Fourier's constants.

Coefficient	**Frame 1**		**Frame 14**		**Frame 21**	
	Upper surface	Lower surface	Upper surface	Lower surface	Upper surface	Lower surface
a_0	11.06	-4.249	7.753	-8.962	157.2	-19.45
a_1	4.702	-12.74	5.979	-6.336	-46.89	8.746
a_2	0.09625	6.147	-3.697	2.854	-210.6	-2.478
a_3	-0.2032	-2.498	1.928	-0.6269	29.98	-0.1496
a_4	-0.1349	0.4049	-0.6374	-0.3204	50.61	0.5221
a_5	-0.01439	0.09634	0.1024	0.2771	-5.49	-0.2742
a_6	0.05231	-0.1105	0.01691	-0.09244	-2.076	-0.02947
b_1	-0.6053	4.657	-4.307	2.373	279.6	2.032
b_2	-0.05436	0.01053	0.4255	0.9047	-41.76	-1.541
b_3	-0.1146	-1.833	0.7825	-1.493	-119.7	1.365
b_4	-0.1641	1.256	-0.7667	0.6059	15.63	-0.4167
b_5	-0.02163	-0.5205	0.458	-0.128	14.55	-0.1122
b_6	0.03344	0.09837	-0.1259	-0.04532	-1.077	0.103
f	0.039	0.03066	0.02889	0.03137	0.0209	0.03097